Wireshark 用户手册

wizardforcel

Published with GitBook

目錄

Wireshark 用户手册

来源：Wiresahrk用户手册

Wireshark 用户手册

介绍Wireshark安装、界面、基本操作

Rechard Sharp

Ed Warnicke

Ulf Lamping

译者序

目录

1. 为什么要翻译

没有特别的目的，2006年年末的时候想找个东西来抓包，找了半天发现了这个叫Wireshark的免费的玩意，当时想翻译一下主页的FAQ，发现FAQ有10几页的样子，玩了几天就放弃了。

2007年以后找到了Wireshark手册，当时想花一两个星期翻译一下试试。于是就开始了

2. 关于本手册

2.1. 用什么工具编写的

开始的时候是用Word写的(虽然知道html,想用html排版超出我的能力)，在翻译前言的时候，手册有提到本书是用Docbook写的，当时顺便提到了，并未尝试使用docbook。后来发现word虽然可以控制章节目录，但是所见即所得对各元素控制能力虽然比较强，但篇幅下来实在是意见困难的事情。最后只有临阵磨枪，学习DOCBOOk。从基本了解到找到合适的工具(oXygen ,xxm)，大约花了1个多星期，逐渐熟悉顺手又花了一个多星期。

也算是收获吧，通过翻译，顺便学会了Docbook.如果有兴趣的可以找找相关的介绍了解一下

本书是使用XMLmind_XML_Editor标准版编写(近似所见即所得)，使用oxygen组织输出的。前者是免费的，有功能限制，后者是D版，组织文档、输出功能比较完善，但编辑文档对我这样的业余水平有点为难。网上有很多介绍自己配置输出环境的，如果我的方法有违自己价值观的，也可以使用前者加上自己配置的输出环境。

2.2. 手册翻译的效果

效果那是相当的差，原因有三

- E文水平差，四级一直没有过。翻译时常会翻一些低级错误，不能把握手册类书籍的翻译风格，比较恶心类似于FAQ的一些常见的句法。。。
- 缺乏对窗口，界面E文名称的了解，出现了大量常识性错误，比如，有很多地方的Frame是指对话框中的一些选项组，我错以为是以太网帧，希望看手册的兄弟留点心，不要被我误导。
- 虽然看过协议方面的书籍，但那些术语都是中文版的，E文版的看的好，术语翻译很多都是望文生义。再次建议大家发扬拿来主义，不要被我误导。

2.3. 翻译中的问题

译文缺少附录部分(实在是没兴趣翻译了)，但是又插入了相关链接，看到类似于???的，要么是链接错误，要么是链接到附录部分了。本想翻译完以后做一番校对，发现输出以后居然有196页，我就彻底放弃了。翻译的常见问题如下：

- 有些窗口名，菜单名本不该翻译的被翻译了.在没有合适中文版的情况下，翻译这些东西会引起用户的不适，开始未曾注意，后来发现了这个问题，部分名称都是E文名/中文名形式，但不是全部。
- 前后不一致问题，有些地方前面用一个词，后面又换成了另外一个词，比如第九章的preferenc,本应该是参数，在译文中有时候是参数，有时候是首选项，请多加注意。
- 很多句子无法理解，稀里糊涂的被翻译了，后面多是直接引用原文，如果发现某些句子难以理解，建议直接查看原文。
- 翻译风格问题：尽量依照原文翻译，大部分风格都和原文类似，一些细小的部分因为自己技巧问题，没有办法实现和原文一致。

3. 补充说明

在翻译快结束的时候，发现网上有peterhu318 网友写的Ethereal手册（中文版），应该是2006年6月6日发布的。Ethereal是Wireshark的前身，想必内容大同小异。郁闷早知道就不写了。

书中有些类容是0.99.5版并未提供的功能，不知道是平台原因，还是功能被丢弃、还是未来功能。

我的邮箱是newguyer[AT]gmail.com,没有精力对本文进行修改了，有兴趣的可以发邮件给我索取DOCbook格式的文件，自行修改补充。

前言

目录

1. 序言

Wireshark 是一种适合网络管理员使用的程序。但因为一直缺乏足够的文档资料阻碍了它的广泛流传。 提高Wireshark易用性，Wireshark团队作出诸多努力，本书即是其中的的一部分。

我们希望本书能对您有所帮助，同时希望您能提出宝贵意见

2. 本书的阅读对象

任何人只要有兴趣或者有需要都可以阅读本书！

本书将会讲解Wireshark的所有基本功能以及一些特性。 与原先的版本相比，今天Wireshark已经拥有非常丰富的功能，当然也变得更加复杂了。也许有些功能在本书中会略过不做讲解。！

本书不尝试揭示网络嗅探(Network sniffing)的原理（译者注：此处翻译不尽人意）。 很多有用的信息可以在Wireshark的WiKI网站上找到 http://wiki.wireshark.org [1]

通过阅读本书，你将会了解如何安装Wireshark，如何使用图形界面的各单元（比如菜单），以及一些不能直接发现的高级功能。本书将会初学者（有时甚至一些高手）解决使用中碰到的一些常见问题.

[1] 译者注：个人认为该网站还比较简陋

3. 感谢

答谢的，此处就不译了,附原文如下

The authors would like to thank the whole Wireshark team for their assistance. In particular, the authors would like to thank:

- Gerald Combs, for initiating the Wireshark project and funding to do this documentation.
- Guy Harris, for many helpful hints and a great deal of patience in reviewing this document.
- Gilbert Ramirez, for general encouragement and helpful hints along the way.

The authors would also like to thank the following people for their helpful feedback on this document:

- Pat Eyler, for his suggestions on improving the example on generating a backtrace.
- Martin Regner, for his various suggestions and corrections.
- Graeme Hewson, for a lot of grammatical corrections.

The authors would like to acknowledge those man page and README authors for the Wireshark project from who sections of this document borrow heavily:

- Scott Renfro from whose mergecap man page Section D.7, “mergecap: Merging multiple capture files into one ” is derived.
- Ashok Narayanan from whose text2pcap man page Section D.8, “text2pcap: Converting ASCII hexdumps to network captures ” is derived.
- Frank Singleton from whose README.idl2wrs Section D.9, “idl2wrs: Creating dissectors from CORBA IDL files ” is derived.

4. 文档约定

本书由Wireshark基金会提供经费， Richard Sharpe编写。经ED Warnicke更新，最近大多数更新，重新排版由UlfLamping进行。

本书使用DocBook/XML编写[2]

阅读本书时你会遇到一些特殊的标记

警告

看到警告时你需要加以注意了，可能会有数据丢失发生

注意

指出一些常见错误，或者一些不明显的东西

提示

提示对日常使用Wireshark很有用

[2] 译者注：未来的文档标准，适合书籍，资料之类的发布，有兴趣的可以google搜索 一下，译文不是用这个写的

5. 从哪里可以得到Wireshark

你可以从我们的网站下载最新版本的Wireshark http://www.wireshark.org/download.html.网站上您可以选择适合您的镜像站点。

Wireshark通常在4-8周内发布一次新版本

如果您想获得Wireshark发布的消息通知，你可以订阅Wireshark-announce邮件列表。详见第1.6.4 节 “邮件列表”

6. 反馈

对本书的反馈信息可以通过wireshark-dev[AT]wireshark.org.发送给作者。

第 1 章 介绍

目录

1.1. 什么是Wireshark

Wireshark 是网络包分析工具。网络包分析工具的主要作用是尝试捕获网络包， 并尝试显示包的尽可能详细的情况。

你可以把网络包分析工具当成是一种用来测量有什么东西从网线上进出的测量工具，就好像使电工用来测量进入电信的电量的电度表一样。（当然比那个更高级）

过去的此类工具要么是过于昂贵，要么是属于某人私有，或者是二者兼顾。 Wireshark出现以后，这种现状得以改变。

Wireshark可能算得上是今天能使用的最好的开元网络分析软件。

1.1.1. 主要应用

下面是Wireshark一些应用的举例：

- 网络管理员用来解决网络问题
- 网络安全工程师用来检测安全隐患
- 开发人员用来测试协议执行情况
- 用来学习网络协议

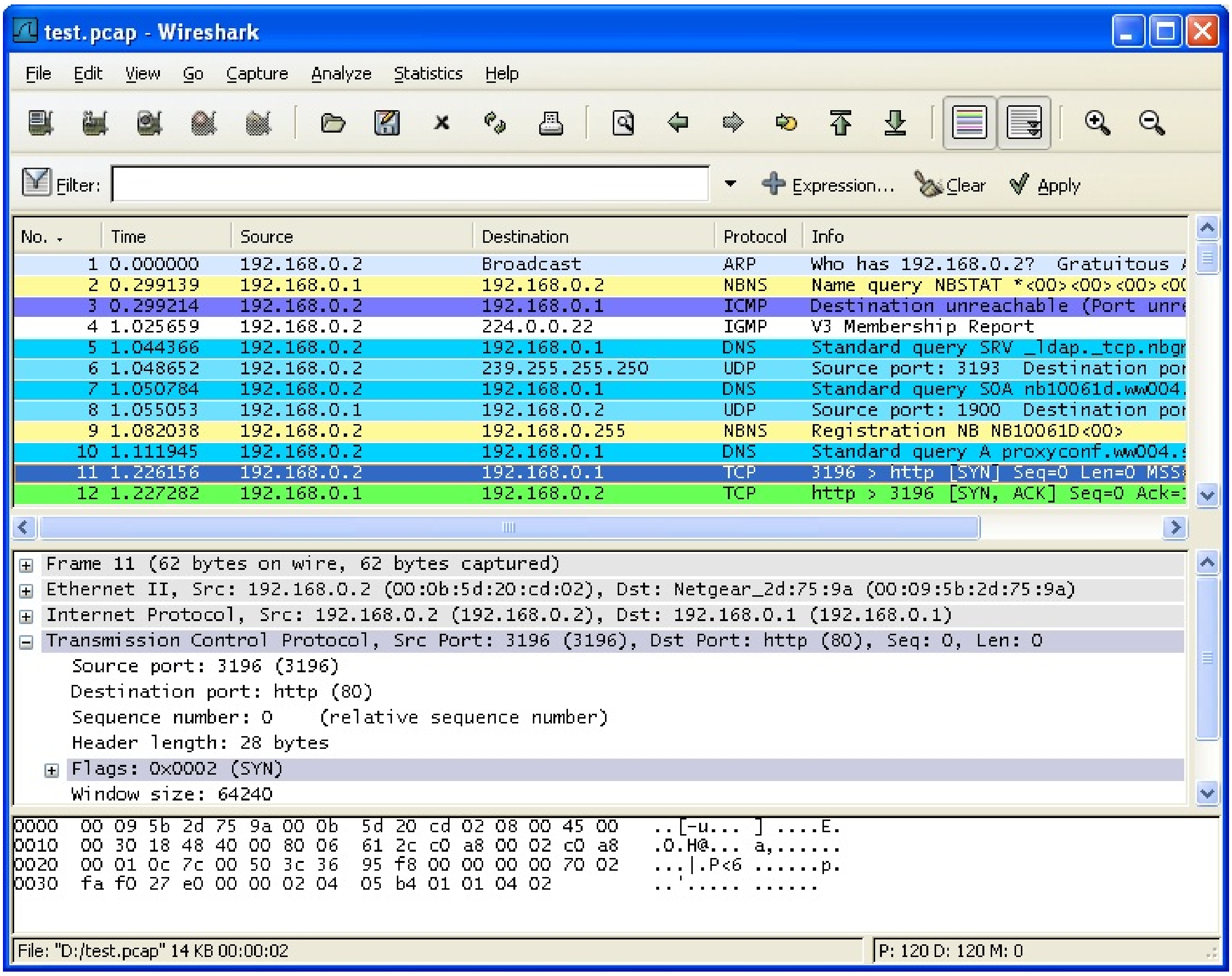
除了上面提到的，Wireshark还可以用在其它许多场合。

1.1.2. 特性

- 支持UNIX和Windows平台
- 在接口实时捕捉包
- 能详细显示包的详细协议信息
- 可以打开/保存捕捉的包
- 可以导入导出其他捕捉程序支持的包数据格式
- 可以通过多种方式过滤包
- 多种方式查找包
- 通过过滤以多种色彩显示包
- 创建多种统计分析
- …还有许多

不管怎么说，要想真正了解它的强大，您还得使用它才行

图 **1.1. Wireshark**捕捉包并允许您检视其内容



1.1.3. 捕捉多种网络接口

Wireshark 可以捕捉多种网络接口类型的包，哪怕是无线局域网接口。想了解支持的所有网络接口类型， 可以在我们的网站上找到http://wiki.wireshark.org/CaptureSetup/NetworkMedia.

1.1.4. 支持多种其它程序捕捉的文件

Wireshark可以打开多种网络分析软件捕捉的包，详见???

1.1.5. 支持多格式输出

Wieshark可以将捕捉文件输出为多种其他捕捉软件支持的格式，详见???

1.1.6. 对多种协议解码提供支持

可以支持许多协议的解码(在Wireshark中可能被称为解剖)???

1.1.7. 开源软件

Wireshark是开源软件项目，用GPL协议发行。您可以免费在 任意数量的机器上使用它，不用担心授权和付费问题，所有的源代码在GPL框架下都可以免费使用。因为以上原因，人们可以很容易在Wireshark上添加新的协议，或者将其作为插件整合到您的程序里，这种应用十分广泛。

1.1.8. Wireshark不能做的事

Wireshark不能提供如下功能

- Wireshark不是入侵检测系统。如果他/她在您的网络做了一些他/她们不被允许的奇怪的事情，Wireshark不会警告您。但是如果发生了奇怪的事情，Wireshark可能对察看发生了什么会有所帮助。 [3]
- Wireshark不会处理网络事务，它仅仅是“测量”(监视)网络。Wireshark不会发送网络包或做其它交互性的事情（名称解析除外，但您也可以禁止解析）。

[3] 译者注：因为不是入侵检测之用，所以不会将入侵检测和普通通信区别对待，但是都会体现在网络包里面，如果您有足够的经验，或许能通过监视网络包发现入侵检测

1.2. 系通需求

想要安装运行Wireshark需要具备的软硬件条件...

1.2.1. 一般说明

- 给出的值只是最小需求，在大多数网络中可以正常使用，但不排除某些情况下不能使用。 [4]
- 在繁忙的网络中捕捉包将很容塞满您的硬盘！举个简单的例子：在100MBIT/s全双工以太网中捕捉数据将会产生750MByties/min的数据！在此类网络中拥有高速的CPU，大量的内存和足够的磁盘空间是十分有必要的。
- 如果Wireshark运行时内存不足将会导致异常终止。可以在http://wiki.wireshark.org/KnownBugs/OutOfMemory察看详细介绍以及解决办法。
- Wireshark作为对处理器时间敏感任务，在多处理器/多线程系统环境工作不会比单独处理器有更快的速度，例如过滤包就是在一个处理器下线程运行，除了以下情况例外：在捕捉包时“实时更新包列表”，此时捕捉包将会运行在一个处理下，显示包将会运行在另一个处理器下。此时多处理或许会有所帮助。[5]

1.2.2. Microsoft Windows

- Windows 2000,XP Home版,XP Pro版,XP Tablet PC，XP Media Center, Server 2003 or Vista(推荐在XP下使用)
- 32-bit奔腾处理器或同等规格的处理器（建议频率：400MHz或更高）,64-bit处理器在WoW64仿真环境下-见一般说明
- 128MB系统内存（建议256Mbytes或更高）
- 75MB可用磁盘空间（如果想保存捕捉文件，需要更多空间） 800*600*（建议*1280*1024或更高）分辨率最少65536(16bit)色，(256色旧设备安装时需要选择”legacy GTK1”)
- 网卡需求：
    - 以太网：windows支持的任何以太网卡都可以
    - 无线局域网卡：见MicroLogix support list, 不捕捉802.11包头和无数据桢。
    - 其它接口见：http://wiki.wireshark.org/CaptureSetup/NetworkMedia

说明

- 基于以下三点原因，将不会对旧版Windows提供支持：没有任何开发人员正在使用那些操作系统，这将使支持变得更加困难，Wireshark运行所依赖的库文件（如GTK，WinPCap等）也放弃对它们的支持。 同样，微软也放弃了对它们的技术支持。
- Windows 95,98和ME不能运行Wireshark。已知的最后一个可以运行在以上平台的版本是Ethereal0.99.0(需要安装WinPCap3.1),你依然可以使用从: http://ethereal.com/download.html获得。顺便提一下：微软于2006年1月11日停止对98/ME支持。
- Windows NT 4.0今后将无法运行Wireshark.最有一个已知版本是Wireshark0.99.4(需安装自带的WinPCap3.1),你依然可以从：http://prdownloads.sourceforge.net/wireshark/wireshark-setup-0.99.4.exe得到它。顺便提一下：微软于2005年12月31日停止对NT 4.0的支持。
- Windows CE 及嵌入版windows（NT/XP）不被支持。
- 64-bit处理器运行Wireshark需要在32bit仿真环境下(称作WoW64),最低需要安装WinPCap4.0。
- 支持多显示(不知道是显示其还是监视器)安装，但会遇到一些不可预料的问题。

1.2.3. Unix/Linux

Wireshark目前可以运行在许多UNIX平台，系统可以对照上面Windows下的指标。 二进制包最少在以下平台可用：

- APPle Mac OSX

- Debian GNU/Linux
- FreeBSD
- NetBSD
- OpenPKG
- Red Hat Fedora/Enterprise Linux
- rPath Linux
- Sun Solaris/i386
- Sun Solaris/Sparc

如果二进制包在您的平台无法使用，你可以下载源文件并尝试编译它。 希望您能发送邮件到wireshark-dev[AT]wireshark.org .分享您的经验。

[4] 译者注：原文 “The values below are the minimum requirements and only "rules of thumb" for use on a moderately used network”，其中”rules of thumb”中译名应该是拇指规则，但网上关于拇指规则解释莫衷一是，大致意思是说：大多数情况下适用，但并非所有情况。这里翻译的有点别扭

[5] 译者注：我对这句话的理解是，正如播放电影一样，高性能的处理器只会增强显示效果，您并不需要将原来30分钟的影片10分钟之内看完。当然，对减少延时还是有作用的。但是感觉这句有点阅读困难，可能翻译的有点问题.

1.4. Wiresahrk简史[6]

1997年以后，Gerald Combs 需要一个工具追踪网络问题并想学习网络知识。所以他开始开发Ethereal (Wireshark项目以前的名称) 以解决以上的两个需要。

Ethereal是第一版，经过数次开发，停顿，1998年，经过这么长的时间，补丁，Bug报告，以及许多的鼓励，0.2.0版诞生了。Ethereal就是以这种方式成功的。

此后不久，Gilbert Ramirez发现它的潜力，并为其提供了底层分析

1998年10月，Guy Harris正寻找一种比TcpView更好的工具，他开始为Ethereal进行改进，并提供分析。

998年以后，正在进行TCP/IP教学的Richard Sharpe 关注了它在这些课程中的作用。并开始研究该软件是否他所需要的协议。如果不行，新协议支持应该很方便被添加。所以他开始从事Ethereal的分析及改进。

从那以后，帮助Ethereal的人越来越多，他们的开始几乎都是由于一些尚不被Ethereal支持的协议。所以他们拷贝了已有的解析器，并为团队提供了改进回馈。

2006年项目Moved House（这句不知道怎么翻译）并重新命名为：Wireshark.

[6] 本段因为有很多协议，程序开发方面的术语，翻译得比较糟糕

1.5. Wireshark开发维护

Wireshark最初由Gerald Combs开发。目前由Wireshark team进行进一步开发和维护。Wireshark team是一个由修补bug提高Wireshark功能的独立成员组成的松散组织。

有大量的成员为Wireshark提供协议分析。同时我们也希望这些活动能持续机芯。通过查看Wireshark帮助菜单下的About,你可以找到为Wireshark提供代码的人员名单，或者你也可以通过Wireshark 网站的authors页面找到。

Wireshark 是开源软件项目，发布遵循GNU General Public Licence (GPL协议),所有源代码可以在GPL框架下免费使用。欢迎您修改Wireshark以便适合您的需要，如果您可以提供您的改进给Wireshark team，我们将不胜感激。

为Wireshark Team 提供您的改进建议，有以下益处：

- 如果其他人发现您提供的改进十分有用会肯定它们的价值，您将会得知你曾像Wireshark team 一样帮助过他人
- The developers of Wireshark might improve your changes even more, as there's always room for improvement. Or they may implement some advanced things on top of your code, which can be useful for yourself too.
- The maintainers and developers of Wireshark will maintain your code as well, fixing it when API changes or other changes are made, and generally keeping it in tune with what is happening with Wireshark. So if Wireshark is updated (which is done often), you can get a new Wireshark version from the website and your changes will already be included without any effort for you.

Wireshar 源代码和二进制kits （二进制工具包？）可以根据自己的平台对应下载，网站是：http://www.wireshark.org/download.html.

1.6. 汇报问题和获得帮助

如果您在使用中碰到了问题，或者您需要Wireshark的帮助，有以下几种可能让您有兴趣的方法（当然，还包括这本书）。

1.6.1. 网站

通过访问http://www.wireshark.org你将会发现关于Wireshark许多有用的信息。

1.6.2. 百科全书

Wireshark Wiki (http://wiki.wireshark.org)提供广泛的跟Wireshark以及捕捉包有关信息。你将会发现一些没有被包括在本书内信息，例如：wiki上有解释如何在交换网络捕捉包，同时我们正努力建立协议参考，等等。

最好的事情是，如果对某些知识有独到见解（比如您精通某种协议），您可以通过浏览器编辑它。

1.6.3. FAQ

最经常被问到的问题“Frequently Asked Questions”提供一个经常被问到的问题以及答案的列表。

> Read The FAQ
>
> 在您发送任何邮件到邮件列表之前，确信您已经阅读了FAQ，因为这里面很可能已经提供了您想问的问题，答案。这将大大节约您的时间（记住，有很多人提交了大量的邮件）。

1.6.4. 邮件列表

下面的几个几个邮件列表，分别属于不同的主题：

Wireshark-users

这是一个Wireshark用户的列表，大家提交关于安装和使用Wireshark的问题，其它人（非常有用）提供的答案。（译者注：其他人当然也是指用户？）

wireshark-announce

这是一个关于程序发布信息的列表，通常每4-8周出现一次。

wireshark-dev

这是一个关于Wireshark开发的邮件列表，如果开始开发协议分析，可以从加入该列表

你可以通过网站http://www.wireshark.org订阅每个邮件列表.简单点击网站左手边的邮件列表链接就可以。邮件同样在网站上可以看到存档。

提示

你可以搜索存档看看有没有人问过跟你一样的问题，或许您的问题已经有了答案。这样您就不必提交邮件以等待别人答复您了。

1.6.5. 报告问题

注意

在您提交任何问题之前，请确定您安装的是最新版本的Wireshark。

当您提交问题的时候，如果您提供如下信息将会对解决问题很有帮助。

1. Wireshark的版本，及其依赖的库的版本，如GTK+，等等。你可以通过**Wireshark –v**命令获得版本号。（估计是UNIX/Linux平台）。
2. 运行Wireshark的平台信息。
3. 关于问题的详细描述。
4. 如果您得到错误或者警告信息，拷贝错误信息的文本（以及在此之前或之后的文本，如果有的话），这样其他人可能会发现发生问题的地方。请不要发送诸如："I got a warning while doing x" [7]，因为这样看起来不是个好主意。

不要发送大文件

不要发送过大的文件（>100KB）到邮件列表，在邮件中附加一个能提供足够数据的记事本就可以。大文件会让很多邮件列表里的那些对您的问题不感兴趣的用户感到恼怒。如果需要，你可以单独发送那些数据给对您问题真正感兴趣，要求您发送数据的人。

不要发送机密信息！

如果您发送捕捉数据到邮件列表，请确定它们不包含敏感或者机密信息，比如密码或者诸如此类的。

1.6.6. 在UNIX/Linux平台追踪软件错误

如果您发送捕捉数据到邮件列表，请确定它们不包含敏感或者机密信息，比如密码或者诸如此类的。

你可以通过如下命令获得追踪信息：

```
$ gdb `whereis wireshark | cut -f2 -d: | cut -d' ' -f2` core >&bt.txt
backtrace
^D
$
```

注意

在逐字输入第一行的字符！[8]

注意

追踪是一个GDB命令。你可以在输完第一上以后输入它，但是会没有相应，^D命令（CTL+D）将会退出GDB命令。以上命令让你在当前目录得到一个名为 `bt.txt` 的文本文件，它包含您的bug报告。

注意

如果您缺少GDB，您必须检查您的操作系统的调试器。

你可以发送追踪邮件到wireshark-dev[AT]wireshark.org邮件列表

1.6.7. 在Windows平台追踪软件错误

Windows下无法包含符号文件 `(.pdb)`,它们非常大。因此不太可能创建十分有意义的追踪文件。你将汇报软件错误就像前面描述的其他问题一样。（这句不尽人意）

[7] 译者注：那句话的意思是，我在XX时碰到一个警告信息

[8] 译者注：原文是："Type the characters in the first line verbatim! Those are back-tics there!",Those are back-tics there!不知道是什么意思，back-tics=后勤抽搐？熟悉Linux的或许知道

第 2 章 编译/安装Wireshark

目录

2.1. 须知

万事皆有开头，Wireshark也同样如此。要想使用Wireshark，你必须：

- 获得一个适合您操作系统的二进制包，或者
- 获得源文件为您的操作系统编译。

目前，只有两到三种Linux发行版可以传送Wireshark，而且通常传输的都是过时的版本。至今尚未有UNIX版本可以传输Wireshark . Windows的任何版本都不能传输Wireshark.基于以上原因，你需要知道从哪能得到最新版本的Wireshark以及如何安装它。

本章节向您展示如何获得源文件和二进制包，如何根据你的需要编译Wireshark源文件。

以下是通常的步骤：

1. 下载需要的相关包，例如：源文件或者二进制发行版。

2. 将源文件编译成二进制包(如果您下载的是源文件的话)。这样做做可以整合编译和/或安装其他需要的包。

3. 安装二进制包到最终目标位置。

2.2. 获得源

你可以从Wireshark网站http://www.wireshark.org.同时获取源文件和二进制发行版。选择您需要下载的链接，然后选择源文件或二进制发行包所在的镜像站点（尽可能离你近一点的站点）。

> 下载所有需要的文件！
>
> 一般来说，除非您已经下载Wireshark,如果您想编译Wireshark源文件，您可能需要下载多个包。这些在后面章节会提到。
>
> 注意
>
> 当你发现在网站上有多个二进制发行版可用，您应该选择适合您平台的版本，他们同时通常会有多个版本紧跟在当前版本后面，那些通常时拥有那些平台的用户编译的。

基于以上原因，您可能想自己下载源文件自己编译，因为这样相对方便一点。

2.3. 在UNIX下安装之前

在编译或者安装二进制发行版之前，您必须确定已经安装如下包：

1. GTK+, The GIMP Tool Kit.

   您将会同样需要Glib.它们都可以从www.gtk.org获得。

2. Libpcap , Wireshark用来捕捉包的工具

   您可以从www.tcpdump.org获得。

根据您操作系统的不同，您或许能够安装二进制包，如RPMs.或许您需要获得源文件并编译它。

如果您已经下载了GTK+源文件，例 2.1 “从源文件编译GTK+”提供的指令对您编译有所帮助。

例 2.1. 从源文件编译GTK+

```
gzip -dc gtk+-1.2.10.tar.gz | tar xvf -
```

```
<much output removed>
```

```
./configure
```

```
<much output removed>
```

```
make install
```

```
<much output remove>
```

```
test--------------
```

> 注意
>
> 您可能需要修改例 2.1 “从源文件编译GTK+”中提供的版本号成对应您下载的GTK+版本。如果GTK的目录发生变更，您同样需要修改它。，tar xvf 显示您需要修改的目录。
>
> 注意
>
> 如果您使用Linux,或者安装了GUN **tar**，您可以使用**tar zxvfgtk+-1.2.10.tar.gz**命令。同样也可能使用**gunzip –c**或者**gzcat**而不是许多UNIX中的**gzip –dc**
>
> 注意
>
> 如果您在windows中下载了gtk+ 或者其他文件。您的文件可能名称为：`gtk+-1_2_8_tar.gz`

如果在执行例 2.1 “从源文件编译GTK+”中的指令时有错误发生的话，你可以咨询GTK+网站。

如果您已经下载了libpcap源，一般指令如例 2.2 “编译、安装libpcap” 显示的那样会帮您完成编译。同样，如果您的操作系统不支持**tcpdump**,您可以从tcpdump网站下载安装它。

例 **2.2.** 编译、安装**libpcap**

```
gzip -dc libpcap-0.9.4.tar.Z | tar xvf -
```

```
<much output removed>
```

```
cd libpcap-0.9.4
```

```
./configure
```

```
<much output removed>
```

```
make
```

```
<much output removed>
```

```
make install
```

```
<much output removed>
```

> 注意
>
> Libpcap的目录需要根据您的版本进行修改。**tar xvf**命令显示您解压缩的目录。

RedHat 6.x及其以上版本环境下（包括基于它的发行版，如Mandrake）,您可以直接运行RPM安装所有的包。大多数情况下的Linux需要安装GTK+和Glib.反过来说，你可能需要安装所有包的定制版。安装命令可以参考例 2.3 “在RedHat Linux 6.2或者基于该版本得发行版下安装需要的RPM包”。如果您还没有安装，您可能需要安装需要的RPMs。

例 **2.3.** 在**RedHat Linux 6.2**或者基于该版本得发行版下安装需要的**RPM**包

```
cd /mnt/cdrom/RedHat/RPMS
```

```
rpm -ivh glib-1.2.6-3.i386.rpm
```

```
rpm -ivh glib-devel-1.2.6-3.i386.rpm
```

```
rpm -ivh gtk+-1.2.6-7.i386.rpm
```

```
rpm -ivh gtk+-devel-1.2.6-7.i386.rpm
```

```
rpm -ivh libpcap-0.4-19.i386.rpm
```

> 注意
>
> 如果您使用RedHat 6.2之后的版本，需要的RMPs包可能已经变化。您需要使用正确的RMPs包。

在Debian下您可以使用apt-ge命令。apt-get 将会为您完成所有的操作。参见例 2.4 “在Deban下安装Deb”

例 **2.4.** 在**Deban**下安装**Deb**

```
apt-get install wireshark-dev
```

2.4. 在UNIX下编译Wireshark

如果在Unix操作系统下可以用如下步骤编译Wireshark源代码：

1. 如果使用Linux则解压**gzip'd tar**文件,如果您使用UNIX，则解压GUN **tar**文件。对于Linux命令如下：

   ```
   tar zxvf wireshark-0.99.5-tar.gz
   ```

   对于 UNIX版本，命令如下

   ```
   gzip -d wireshark-0.99.5-tar.gz
   ```

   ```
   tar xvf wireshark-0.99.5-tar
   ```

   | | 注意 | |

   使用管道命令行 gzip –dc Wireshark-0.99.5-tar.gz|tar xvf 同样可以[9]

   |

| | 注意 | |

如果您在Windows下下载了Wireshark,你会发现文件名中的那些点变成了下划线。

|

2. 将当前目录设置成源文件的目录。
3. 配置您的源文件以编译成适合您的Unix的版本。命令如下：

```
./configure
```

如果找个步骤提示错误，您需要修正错误，然后重新configure.解决编译错误可以参考第 2.6 节 “解决UNIX下安装过程中的问题 ”

4. 使用make命令将源文件编译成二进制包，例如：

```
make
```

5. 安装您编译好的二进制包到最终目标，使用如下命令：

```
make install
```

一旦您使用make install安装了Wireshark,您就可以通过输入Wireshark命令来运行它了。

[9] 译者注：看到别人翻译Pipelin之类的，似乎就是叫管道，不知道是否准确

2.5. 在UNIX下安装二进制包

一般来说，在您的UNIX下安装二进制发行包使用的方式根据您的UNIX的版本类型而各有不同。例如AIX下，您可以使用smit安装，Tru64 UNIX您可以使用 setld 命令。

2.5.1. 在Linux或类似环境下安装RPM包

使用如下命令安装Wireshark RPM包

```
rpm -ivh wireshark-0.99.5.i386.rpm
```

如果因为缺少Wireshark依赖的软件而导致安装错误，请先安装依赖的软件，然后再尝试安装。REDHAT下依赖的软件请参考例 2.3 “在RedHat Linux 6.2或者基于该版本得发行版下安装需要的RPM包”

2.5.2. 在Debian环境下安装Deb包

使用下列命令在Debian下安装Wireshark

```
apt-get install Wireshark
```

apt-get 会为您完成所有的相关操作

2.5.3. 在Gentoo Linux环境下安装Portage

使用如下命令在Gentoo Linux下安装wireshark以及所有的需要的附加文件

```
USE="adns gtk ipv6 portaudio snmp ssl kerberos threads selinux" emerge wireshark
```

2.5.4. 在FreeBSD环境下安装包

使用如下命令在FreeBSD下安装Wireshark

```
pkg_add -r wireshark
```

pkg_add会为您完成所有的相关操作

2.6. 解决UNIX下安装过程中的问题 [10]

安装过程中可能会遇到一些错误信息。这里给出一些错误的解决办法：

如果**configure**那一步发生错误。你需要找出错误的原因，您可以检查日志文件config.log(在源文件目录下)，看看都发生了哪些错误。有价值的信息通常在最后几行。

一般原因是因为您缺少GTK+环境，或者您的GTK+版本过低。configure错误的另一个原因是因为因为缺少libpcap(这就是前面提到的捕捉包的工具)。

另外一个常见问题是很多用户抱怨最后编译、链接过程需要等待太长时间。这通常是因为使用老式的**sed**命令（比如solaris下传输）。自从libtool脚本使用sed命令建立最终链接命令，常常会导致不可知的错误。您可以通过下载最新版本的sed解决该问题 http://directory.fsf.org/GNU/sed.html.

如果您无法检测出错误原因。发送邮件到wireshark-dev说明您的问题。当然，邮件里要附上config.log以及其他您认为对解决问题有帮助的东西，例如make过程的追踪。

[10] 译者注：本人不熟悉UNIX/LINUX，这一段翻译的有点云里雾里，可能大家通过这部分想安装Wireshark会适得其反，那就对不住了。下面个人说一下UNIX/LINUX下安装方法。UNIX/LINUX下安装时，有两种安装方式，1是下载源码包自己编译，这种方式的好处是因为

下载源码包是单一的，可以自行加以修改，编译就是适合自己平台的了。 2、是利用已经做好的发行包直接安装，这种方法的好处是只要下载到跟自己平台对应的就可以，但缺点也在这里，不是每个平台都能找到合适的。不管是编译安装，还是使用发行包安装，都需要有一些有些基本基本支持。比如Linux下的GTK+支持，捕捉包时需要用的libpcap. 这一点可以参考第 2.3 节 “在UNIX下安装之前 ”。编译的一般步骤是解压，编译，安装（**tar zxvf Wireshark-0.99.5-tar.gz;make;make install**）.直接安装则是根据各自平台安装的特点。

2.7. 在Windows下编译源

在Windows平台下，我们建议最好是使用二进制包直接安装，除非您是从事Wireshark开发的。 如果想了解关于Windows下编译安装Wireshark，请查看我们的开发WIKI网站 http://wiki.wireshark.org/Development来了解最新的开发方面的文档。

2.8. 在Windows下安装Wireshark

本节将探讨在Windows下安装Wireshark二进制包。

2.8.1. 安装Wireshark

您获得的Wireshark二进制安装包可能名称类似 `wireshark-setup-x.y.z.exe.` Wireshark安装包包含WinPcap,所以您不需要单独下载安装它。

您只需要在http://www.wireshark.org/download.html#releases下载Wireshark安装包并执行它即可。除了普通的安装之外，还有几个组件供挑选安装。

> 提示：尽量保持默认设置
>
> 如果您不了解设置的作用的话。

选择组件[11]

Wireshark(包括GTK1和GTK2接口无法同时安装):

如果您使用GTK2的GUI界面遇到问题可以尝试GTK1，在Windows下256色（8bit）显示模式无法运行GTK2.但是某些高级分析统计功能在GTK1下可能无法实现。

- **Wireshark GTK1**-Wireshark 是一个GUI网络分析工具
- **Wireshark GTK2**-Wireshark 是一个GUI网络分析工具（建议使用GTK2 GUI模组工具）
- **GTK-Wimp**-GTKWimp是诗歌GTK2窗口模拟(看起来感觉像原生windows32程序，推荐使用)

- **TSshark**-TShark 是一个命令行的网络分析工具

插件/扩展(Wireshark,TShark分析引擎):

- **Dissector Plugins**-分析插件：带有扩展分析的插件
- **Tree Statistics Plugins**-树状统计插件：统计工具扩展
- **Mate - Meta Analysis and Tracing Engine (experimental)**:可配置的显示过滤引擎，参考http://wiki.wireshark.org/Mate.
- **SNMP MIBs**: SNMP，MIBS的详细分析。

Tools/工具(处理捕捉文件的附加命令行工具

User's Guide-用户手册-本地安装的用户手册。如果不安装用户手册，帮助菜单的大部分按钮的结果可能就是访问internet.

- **Editcap** - Editcap is a program that reads a capture file and writes some or all of the packets into another capture file. /Editcap是一个读取捕捉文件的程序，还可以将一个捕捉文件力的部分或所有信息写入另一个捕捉文件。（文件合并or插入？）
- **Text2Pcap** - Text2pcap is a program that reads in an ASCII hex dump and writes the data into a libpcap-style capture file./Tex2pcap是一个读取ASCII hex，写入数据到libpcap个文件的程序。
- **Mergecap** - Mergecap is a program that combines multiple saved capture files into a single output file. / Mergecap是一个可以将多个播捉文件合并为一个的程序。
- **Capinfos** - Capinfos is a program that provides information on capture files. /Capinfos是一个显示捕捉文件信息的程序。

“Additional Tasks”页

- **Start Menu Shortcuts**-开始菜单快捷方式-增加一些快捷方式到开始菜单
- **Desktop Icon**-桌面图标-增加Wireshark图标到桌面
- **Quick Launch Icon**-快速启动图标-增加一个Wireshark图标到快速启动工具栏
- **Associate file extensions to Wireshark**-Wireshark文件关联-将捕捉包默认打开方式关联到Wireshark

Install WinPcap?”页

Wireshark安装包里包含了最新版的WinPcap安装包。

如果您没有安装WinPcap 。您将无法捕捉网络流量。但是您还是可以打开以保存的捕捉包文件。

- **Currently installed WinPcap version**-当前安装的WinPcap版本
- **Install WinPcap x.x** -如果当前安装的版本低于Wireshark自带的，该选项将会是默认值。
- **Start WinPcap service "NPF" at startup** -将WinPcap的服务NPF在启动时运行-这样其它非管理员用户就同样可以捕捉包了。

更多关于WinPcap的信息：

- Wireshark 相关http://wiki.wireshark.org/WinPcap
- WinPcap官方网站：http://www.winpcap.org

安装命令选项

您可以直接在命令行运行安装包，不加任何参数，这样会显示常用的参数以供交互安装。 在个别应用中，可以选择一些参数定制安装：

- **/NCRC** 禁止CRC校检
- **/S** 静默模式安装或卸载Wireshark.注意：静默模式安装时不会安装WinPcap!
- **/desktopicon** 安装桌面图标，/desktopicon=yes表示安装图标，反之则不是，适合静默模式。
- **/quicklaunchicon** 将图标安装到快速启动工具栏，=yes-安装到工具栏，=no-不安装，不填按默认设置。
- **/D** 设置默认安装目录(`$INSTDIR`),首选安装目录和安装目录注册表键值，该选项必须设置到最后。即使路径包含空格

例 **2.5.**

```
wireshark-setup-0.99.5.exe /NCRC /S /desktopicon=yes /quicklaunchicon=no /D=C:\Program Fi
```

2.8.2. 手动安装WinPcap

注意

事先声明，Wireshark安装时会谨慎对待WinPcap的安装，所以您通常不必担心WinPcap。

下面的WinPcap仅适合您需要尝试未包括在Wireshark内的不同版本WinPcap。例如一个新版本的WinPcap发布了，您需要安装它。

单独的WinPcap版本（包括alpha or beta版）可以在下面地址下载到

- WinPcap官方网站：http://www.winpcap.org
- Wiretapped.net 镜像站点: http://www.mirrors.wiretapped.net/security/packet-capture/winpcap

在下载页面您将会发现WinPcap的安装包名称通常类似于”auto-installer”。它们可以在NT4.0/2000/XP/vista下安装。

2.8.3. 更新Wireshark

有时候您可能想将您的WinPcap更新到最新版本，如果您订阅了Wireshark通知邮件，您将会获得Wireshark新版本发布的通知，见第 1.6.4 节 “邮件列表”

。

新版诞生通常需要8-12周。更新Wireshark就是安装一下新版本。下载并安装它就可以。更新通常不需要重新启动，也不会更改过去的默认设置

2.8.4. 更新WinPcap

WinPcap的更新不是十分频繁，通常一年左右。新版本出现的时候您会收到WinPcap的通知。更新WinPcap后需要重新启动。

> 警告
>
> 在安装新版WinPcap之前，如果您已经安装了旧版WinPcap,您必须先卸载它。最近版本的WinPcap安装时会自己卸载旧版。

2.8.5. 卸载Wireshark

你可以用常见方式卸载Wireshark,使用添加/删除程序，选择”Wireshark”选项开始卸载即可。

Wireshark卸载过程中会提供一些选项供您选择卸载哪些部分，默认是卸载核心组件，但保留个人设置和WinPcap.

WinPcap默认不会被卸载，因为其他类似Wireshark的程序有可能同样适用WinPcap

2.8.6. 卸载WinPcap

你可以单独卸载WinPcap,在添加/删除程序选择”WinPcap”卸载它。

> 注意
>
> 卸载WinPcap之后您将不能使用Wireshark捕捉包。

在卸载完成之后最好重新启动计算机。

[11] 涉及到过多的名次，软件又没有中文版，这里及以后尽量不翻译名称

第 3 章 用户界面

目录

3.1. 须知

现在您已经安装好了Wireshark,几乎可以马上捕捉您的一个包。紧接着的这一节我们将会介绍：

- Wireshark的用户界面如何使用
- 如何捕捉包
- 如何查看包
- 如何过滤包
- ……以及其他的一些工作。

3.2. 启动Wireshark

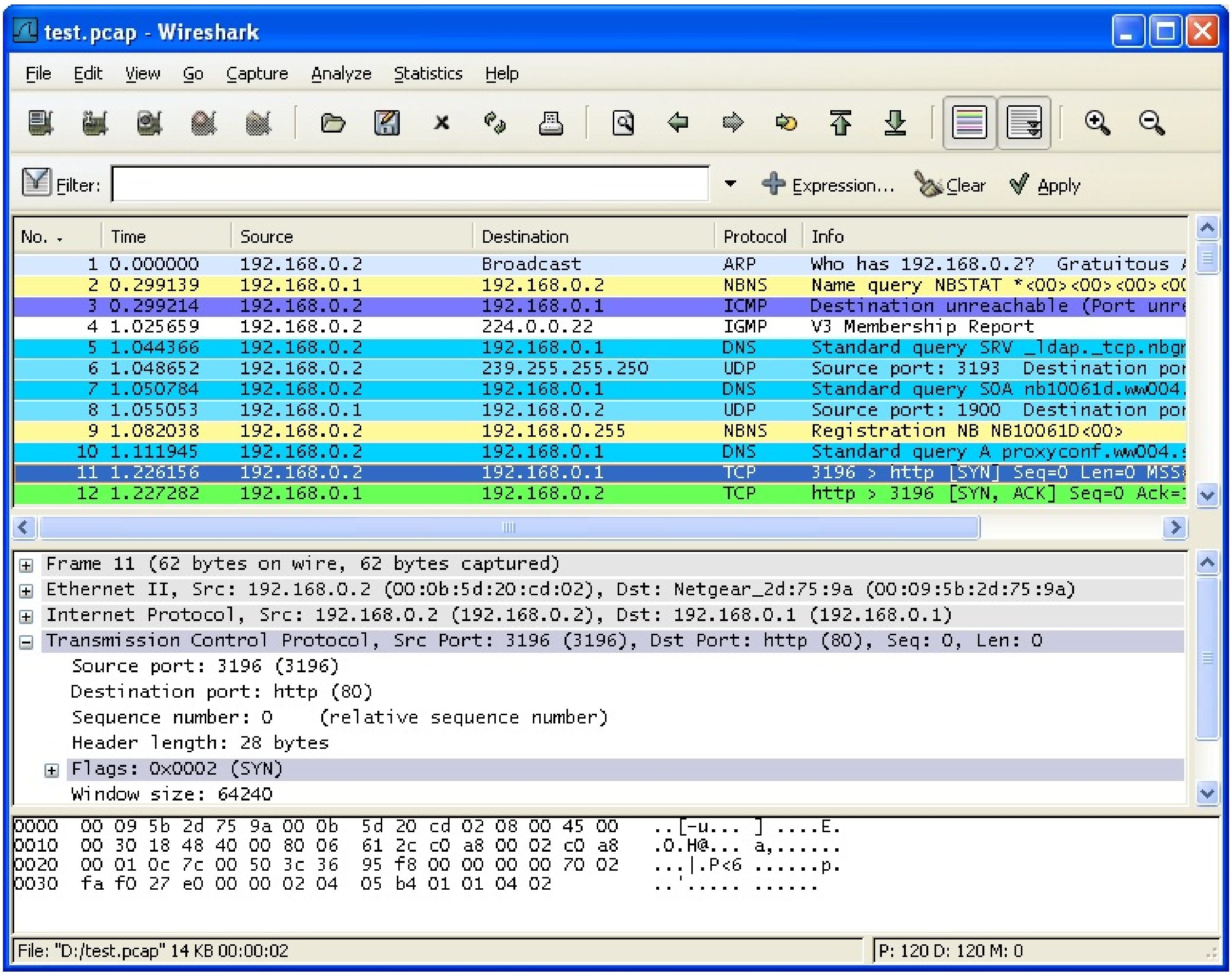
你可以使用Shell命令行或者资源管理器启动Wireshark.

> 提示
>
> 开始Wireshark时您可以指定适当的参数。参见第 9.2 节 “从命令行启动Wireshark”
>
> 注意
>
> 在后面的章节中，将会出现大量的截图，因为Wireshark运行在多个平台， 并且支持多个GUI Toolkit(GTK1.x/2x),您的屏幕上显示的界面可能与截图不尽吻合。但在功能上不会有实质性区别。尽管有这些区别，也不会导致理解上的困难。

3.3. 主窗口

先来看看图 3.1 “主窗口界面”，大多数打开捕捉包以后的界面都是这样子（如何捕捉/打开包文件随后提到）。

图 **3.1.** 主窗口界面



和大多数图形界面程序一样，Wireshark主窗口由如下部分组成：

1. 菜单（见第 3.4 节 “主菜单”）用于开始操作。
2. 主工具栏(见第 3.13 节 “"Main"工具栏”)提供快速访问菜单中经常用到的项目的功能。
3. Fiter toolbar/过滤工具栏(见第 3.14 节 “"Filter"工具栏”)提供处理当前显示过滤得方法。(见6.3:”浏览时进行过滤”)
4. Packet List面板（见第 3.15 节 “"Pcaket List"面板”）显示打开文件的每个包的摘要。点击面板中的单独条目，包的其他情况将会显示在另外两个面板中。
5. Packet detail面板（见第 3.16 节 “"Packet Details"面板”）显示您在Packet list面板中选择的包德更多详情。
6. Packet bytes面板（见第 3.17 节 “"Packet Byte"面板”）显示您在Packet list面板选择的包的数据，以及在Packet details面板高亮显示的字段。
7. 状态栏（见第 3.18 节 “状态栏”）显示当前程序状态以及捕捉数据的更多详情。

注意

主界面的三个面版以及各组成部分可以自定义组织方式。见第 9.5 节 “首选项”

3.3.1. 主窗口概述

Packet list和Detail 面版控制可以通过快捷键进行。表 3.1 “导航快捷键”显示了相关的快捷键列表。表 3.5 “"GO"菜单项”有关于快捷键的更多介绍

表 **3.1.** 导航快捷键

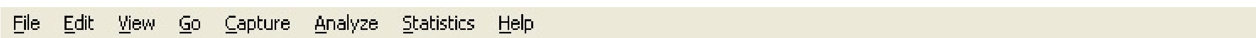
快捷键	描述
Tab,Shift+Tab	在两个项目间移动，例如从一个包列表移动到下一个
Down	移动到下一个包或者下一个详情
Up	移动到上一个包或者上一个详情
Ctrl-Down,F8	移动到下一个包，即使焦点不在Packet list面版
Ctrl-UP,F7	移动到前一个报，即使焦点不在Packet list面版
Left	在Pactect Detail面版，关闭被选择的详情树状分支。如果以关闭，则返回到父分支。
Right	在Packet Detail面版，打开被选择的树状分支.
Backspace	Packet Detail面版，返回到被选择的节点的父节点
Return,Enter	Packet Detail面版，固定被选择树项目。

另外，在主窗口键入任何字符都会填充到filter里面。

3.4. 主菜单

Wireshark主菜单位于Wireshark窗口的最上方。图 3.2 “主菜单”提供了菜单的基本界面。

图 **3.2.** 主菜单



主菜单包括以下几个项目:

File

包括打开、合并捕捉文件，save/保存,Print/打印,Export/导出捕捉文件的全部或部分。以及退出Wireshark项.见第 3.5 节 “"File"菜单”

Edit

包括如下项目：查找包，时间参考，标记一个多个包，设置预设参数。（剪切，拷贝，粘贴不能立即执行。）见第 3.6 节 “"Edit"菜单”

View

控制捕捉数据的显示方式，包括颜色，字体缩放，将包显示在分离的窗口，展开或收缩详情面版的地树状节点，……见第 3.7 节 “"View"菜单”

GO

包含到指定包的功能。见第 3.8 节 “"Go"菜单”

Capture

允许您开始或停止捕捉、编辑过滤器。见第 3.9 节 “"Capture"菜单”

Analyze

包含处理显示过滤，允许或禁止分析协议，配置用户指定解码和追踪TCP流等功能。见第 3.10 节 “"Analyze"菜单”

Statistics

包括的菜单项用户显示多个统计窗口，包括关于捕捉包的摘要，协议层次统计等等。见第 3.11 节 “"Statistics"菜单”

Help

包含一些辅助用户的参考内容。如访问一些基本的帮助文件，支持的协议列表，用户手册。在线访问一些网站，“关于”等等。见第 3.12 节 “"Help"菜单”

本章链接介绍菜单的一般情况，更详细的介绍会出现在后续章节。

> 提示
>
> 你可以直接点击访问菜单项，也可以使用热键，热键显示在菜单文字描述部分。例如：您可以使用CTR+K打开捕捉对话框。

3.5. "File"菜单

WireSharkFile菜单包含的项目如表表 3.2 “File菜单介绍”所示

图 **3.3. File**菜单

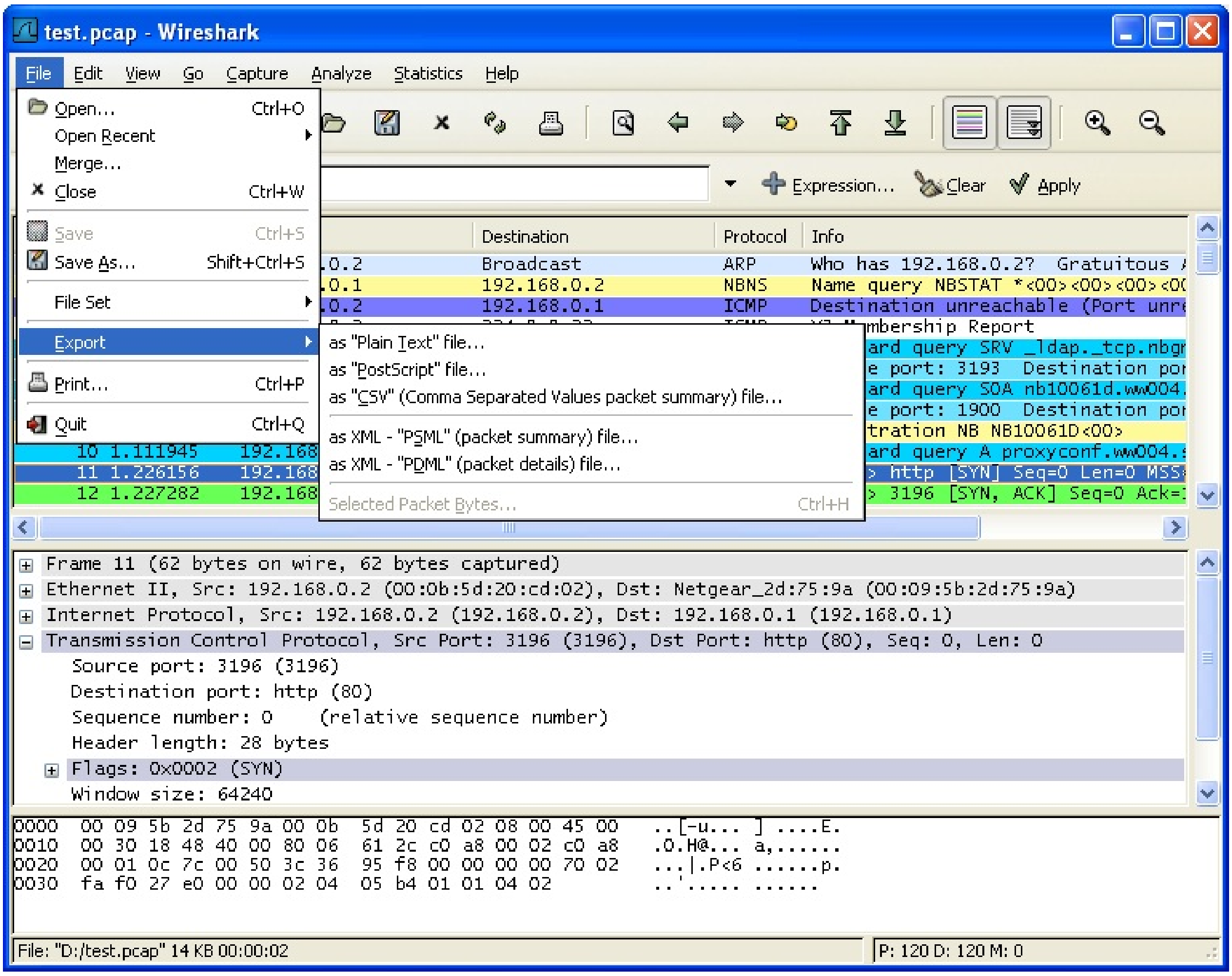


表 3.2. File菜单介绍

菜单项	快捷键	描述
Open...	Ctr+O	显示打开文件对话框，让您載入捕捉文件用以浏览。见第 5.2.1 节 "打开捕捉文件对话框"
Open Recent		弹出一个子菜单显示最近打开过的文件供选择。
Merg		显示合并捕捉文件的对话框。让您选择一个文件和当前打开的文件合并。见第 5.4 节 "合并捕捉文件"
Close	Ctrl+W	关闭当前捕捉文件，如果您未保存，系统将提示您是否保存（如果您预设了禁止提示保存，将不会提示）
Save	Crl+S	保存当前捕捉文件，如果您没有设置默认的保存文件名，Wireshark出现提示您保存文件的对话框。详情第 5.3.1 节 ""save Capture File As/保存文件为"对话框"
		
		注意

		如果您已经保存文件，该选项会是灰色不可选的。
		注意
		您不能保存动态捕捉的文件。您必须结束捕捉以后才能进行保存
Save As	Shift+Ctrl+S	让您将当前文件保存为另外一个文件面，将会出现一个另存为的对话框(参见第 5.3.1 节 “"save Capture File As/保存文件为"对话框”)
File Set>List Files		允许您显示文件集合的列表。将会弹出一个对话框显示已打开文件的列表,参见第 5.5 节 “文件集合”
File Set>Next File		如果当前载入文件是文件集合的一部分，将会跳转到下一个文件。如果不是，将会跳转到最后一个文件。这个文件选项将会是灰色。
File set>Previous Files		如果当前文件是文件集合 的一部分，将会调到它所在位置的前一个文件。如果不是则跳到文件集合的第一个文件，同时变成灰色。
Export> as “Plain Text” File…		这个菜单允许您将捕捉文件中所有的或者部分的包导出为plain ASCII text格式。它将会弹出一个Wireshark导出对话框,见第 5.6.1 节 “"Export as Plain Text File"对话框”
Export >as "PostScript" Files		将捕捉文件的全部或部分导出为PostScrit文件。将会出现导出文件对话框。参见第 5.6.2 节 “"Export as PostScript File" 对话框”
Export > as "CVS" (Comma Separated Values Packet Summary)File...		导出文件全部或部分摘要为.cvs格式（可用在电子表格中）。将会弹出导出对话框,见第 5.6.3 节 “"Export as CSV (Comma Separated Values) File" 对话框”。
Export > as “PSML” File…		导出文件的全部或部分为PSML格式（包摘要标记语言）XML文件。将会弹出导出文件对话框。见第 5.6.4 节 “"Export as PSML File" 对话框”
Export as "PDML" File...		导出文件的全部或部分为PDML(包摘要标记语言)格式的XML文件。将会弹出一个导出文件对话框,见第 5.6.5 节 “"Export as PDML File" 对话框”
Export > Selected Packet Bytes…		导出当前在Packet byte面版选择的字节为二进制文件。将会弹出一个导出对话框。见第 5.6.6 节 “"Export selected packet bytes" 对话框”
		打印捕捉包的全部或部分，将会弹出打印对

Print	Ctrl+P	话框。见第 5.7 节 "打印包"
Quit	Ctrl+Q	退出Wireshark,如果未保存文件，Wireshark会提示是否保存。

3.6. "Edit"菜单

Wireshark的"Edit"菜单包含的项目见表 3.3 "Edit菜单项"

图 3.4. "Edit"菜单

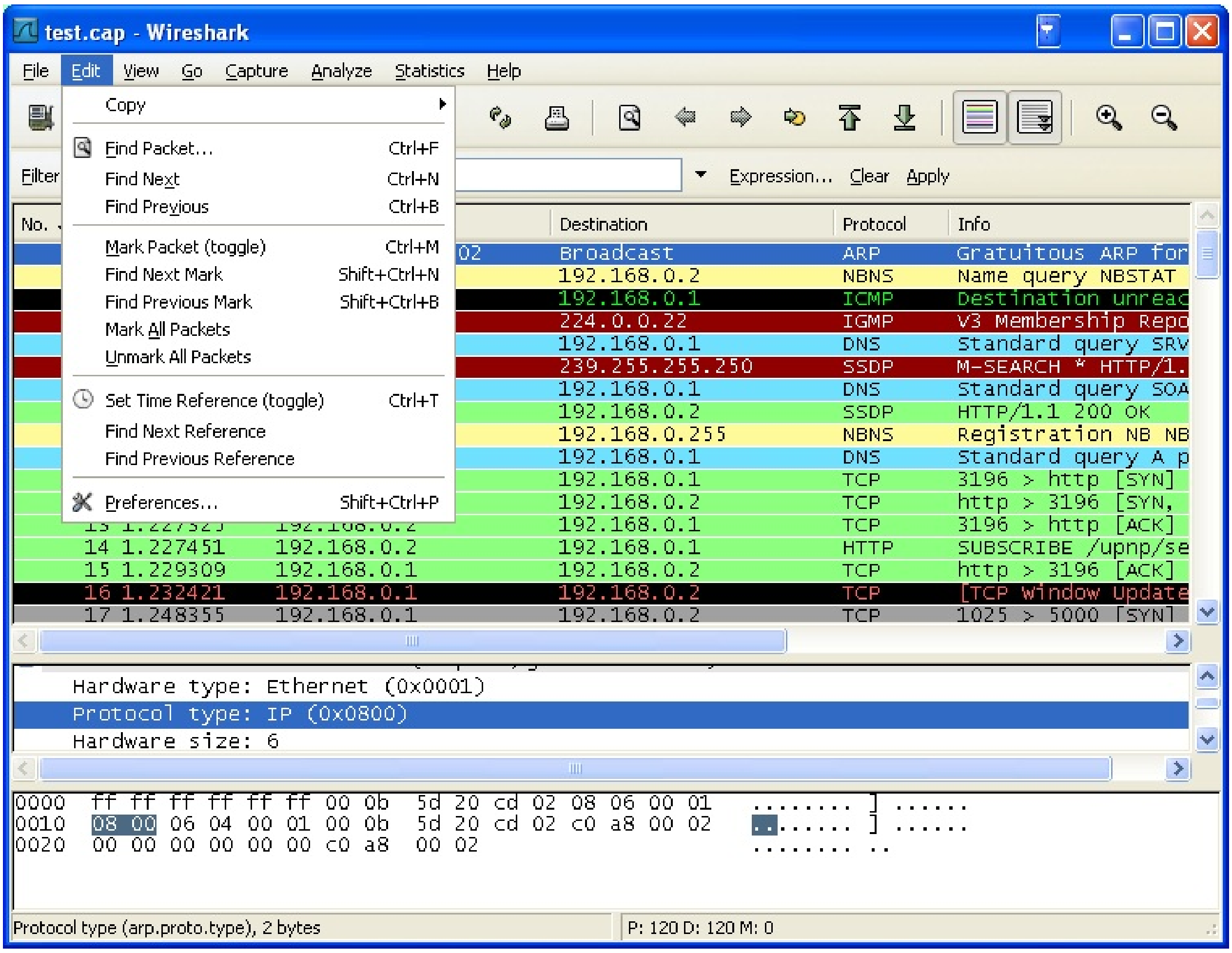


表 3.3. Edit菜单项

菜单项	快捷键	描述
Copy>As Filter	Shift+Ctrl+C	使用详情面版选择的数据作为显示过滤。显示过滤将会拷贝到剪贴板。
Find Packet...	Ctr+F	打开一个对话框用来通过限制来查找包，见???
Find Next	Ctrl+N	在使用Find packet以后，使用该菜单会查找匹配规则的下一个包
Find Previous	Ctr+B	查找匹配规则的前一个包。
Mark Packet(toggle)	Ctrl+M	标记当前选择的包。见第 6.9 节 “标记包”
Find Next Mark	Shift+Ctrl+N	查找下一个被标记的包
Find Previous Mark	Ctrl+Shift+B	查找前一个被标记的包
Mark ALL Packets		标记所有包
Unmark All Packet		取消所有标记
Set Time Reference(toggle)	**Ctrl+T**	以当前包时间作为参考,见第 6.10.1 节 “包参考时间”
Find Next Reference		找到下一个时间参考包
Find Previous Refrence...		找到前一个时间参考包
Preferences...	Shift+Ctrl+P	打开首选项对话框，个性化设置Wireshark的各项参数，设置后的参数将会在每次打开时发挥作用。详见第 9.5 节 “首选项”

3.7. "View"菜单

表 3.4 “"View"菜单项”显示了Wireshar View菜单的选项

图 **3.5. "View"**菜单

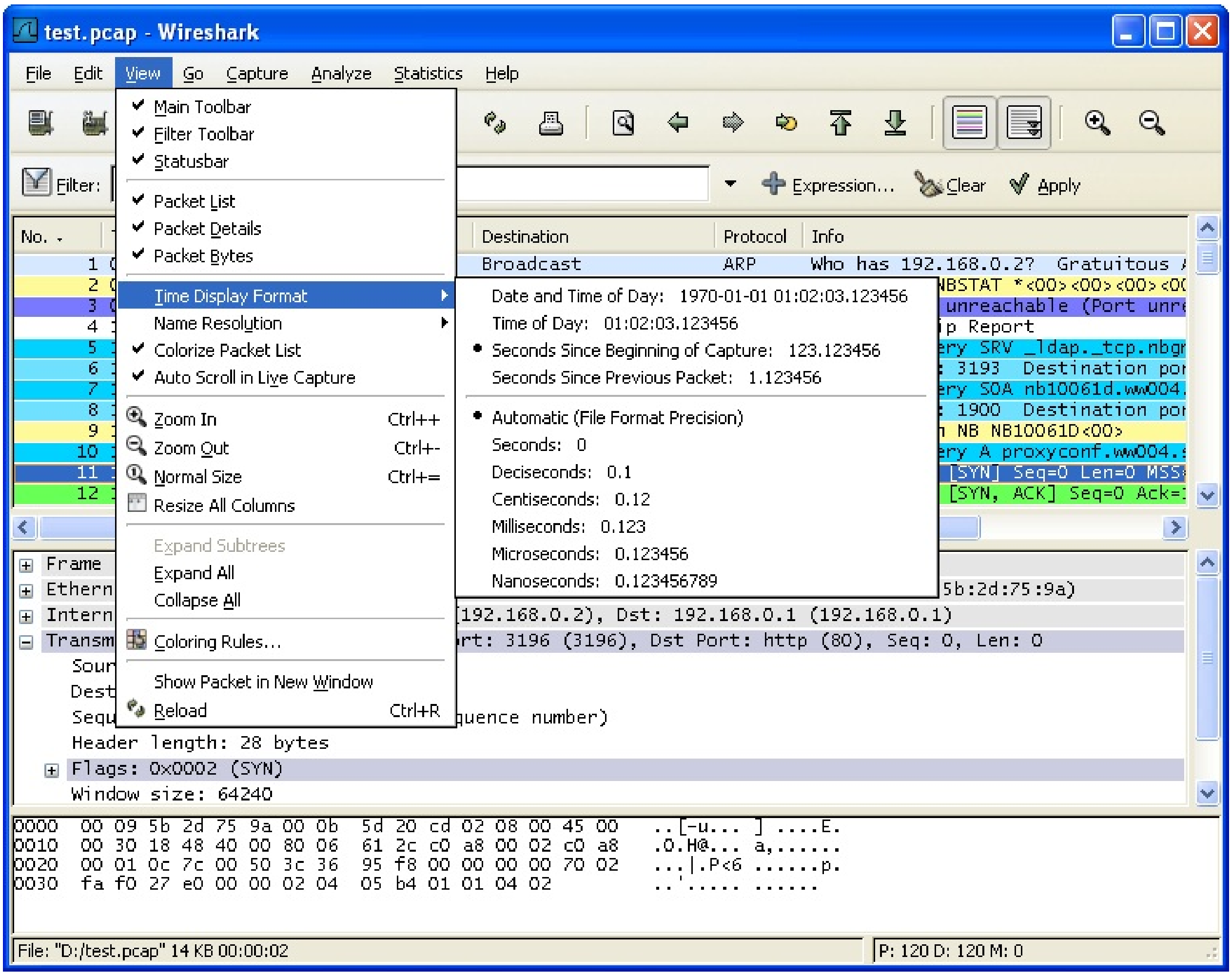


表 3.4. "View"菜单项

菜单项	快捷键	描述
Main Toolbar		显示隐藏Main toolbar(主工具栏),见第 3.13 节 “"Main"工具栏”
Filter Toolbar		显示或隐藏Filter Toolbar(过滤工具栏)见第 3.14 节 “"Filter"工具栏”
Statusbar		显示或隐藏状态栏,见第 3.18 节 “状态栏”
Packet List		显示或隐藏Packet List pane(包列表面板),见第 3.15 节 “"Pcaket List"面板”
Packet Details		显示或隐藏Packet details pane(包详情面板).见第 3.16 节 “"Packet Details"面板”
Packet Bytes		显示或隐藏 packet Bytes pane(包字节面板), 见第 3.17 节 “"Packet Byte"面板”
Time Display Fromat>Date and Time of Day: 1970-01-01 01:02:03.123456		选择这里告诉Wireshark将时间戳设置为绝对日期-时间格式(年月日, 时分秒), 见第 6.10 节 “时间显示格式及参考时间”

		注意
		这里的字段"Time of Day","Date and Time of Day","Seconds Since Beginning of Capture","Seconds Since Previous Captured Packet"和"Seconds Since Previous Displayed Packet"几个选项是互斥的，换句话说，一次同时有一个被选中。
Time Display Format>Time of Day: 01:02:03.123456		将时间设置为绝对时间-日期格式(时分秒格式),见第 6.10 节 “时间显示格式及参考时间”
Time Display Format > Seconds Since Beginning of Capture: 123.123456		将时间戳设置为秒格式，从捕捉开始计时，见第 6.10 节 “时间显示格式及参考时间”
Time Display Format > Seconds Since Previous Captured Packet: 1.123456		将时间戳设置为秒格式，从上次捕捉开始计时，见第 6.10 节 “时间显示格式及参考时间”
Time Display Format > Seconds Since Previous Displayed Packet: 1.123456		将时间戳设置为秒格式，从上次显示的包开始计时,见第 6.10 节 “时间显示格式及参考时间”
Time Display Format > ------		
Time Display Format > Automatic (File Format Precision)		根据指定的精度选择数据包中时间戳的显示方式，见第 6.10 节 “时间显示格式及参考时间”
		注意
		"Automatic","Seconds"和"...seconds"是互斥的
Time Display Format > Seconds: 0		设置精度为1秒，见第 6.10 节 “时间显示格式及参考时间”
Time Display		设置精度为1秒，0.1秒，0.01秒，百万分之一秒等等。

Format > ...seconds: 0....		见第 6.10 节 "时间显示格式及参考时间"
Name Resolution > Resolve Name		仅对当前选定包进行解析第 7.6 节 "名称解析"
Name Resolution > Enable for MAC Layer		是否解析Mac地址
Name Resolution > Enable for Network Layer		是否解析网络层地址(ip地址),见第 7.6 节 "名称解析"
Name Resolution > Enable for Transport Layer		是否解析传输层地址第 7.6 节 "名称解析"
Colorize Packet List		是否以彩色显示包
		注意
		以彩色方式显示包会降低捕捉再如包文件的速度
Auto Scrooll in Live Capture		控制在实时捕捉时是否自动滚屏，如果选择了该项，在有新数据进入时，面板会项上滚动。您始终能看到最后的数据。反之，您无法看到满屏以后的数据，除非您手动滚屏
Zoom In	Ctrl++	增大字体
Zoom Out	Ctrl+-	缩小字体
Normal Size	Ctrl+=	恢复正常大小
Resiz All Columnus		恢复所有列宽
		注意
		除非数据包非常大，一般会立刻更改
Expend Subtrees		展开子分支
Expand All		看开所有分支，该选项会展开您选择的包的所有分支。
Collapse All		收缩所有包的所有分支
Coloring Rulues...		打开一个对话框，让您可以通过过滤表达来用不同的颜色显示包。这项功能对定位特定类型的包非常有用。见第 9.3 节 "包色彩显示设置"
Show Packet in		在新窗口显示当前包，(新窗口仅包含View,Byte View两

New Window		个面板)
Reload	Ctrl+R	重新再如当前捕捉文件

3.8. "Go"菜单

Wireshark "GO"菜单的内容见表 3.5 “"GO"菜单项”

图 **3.6. "GO"菜单**

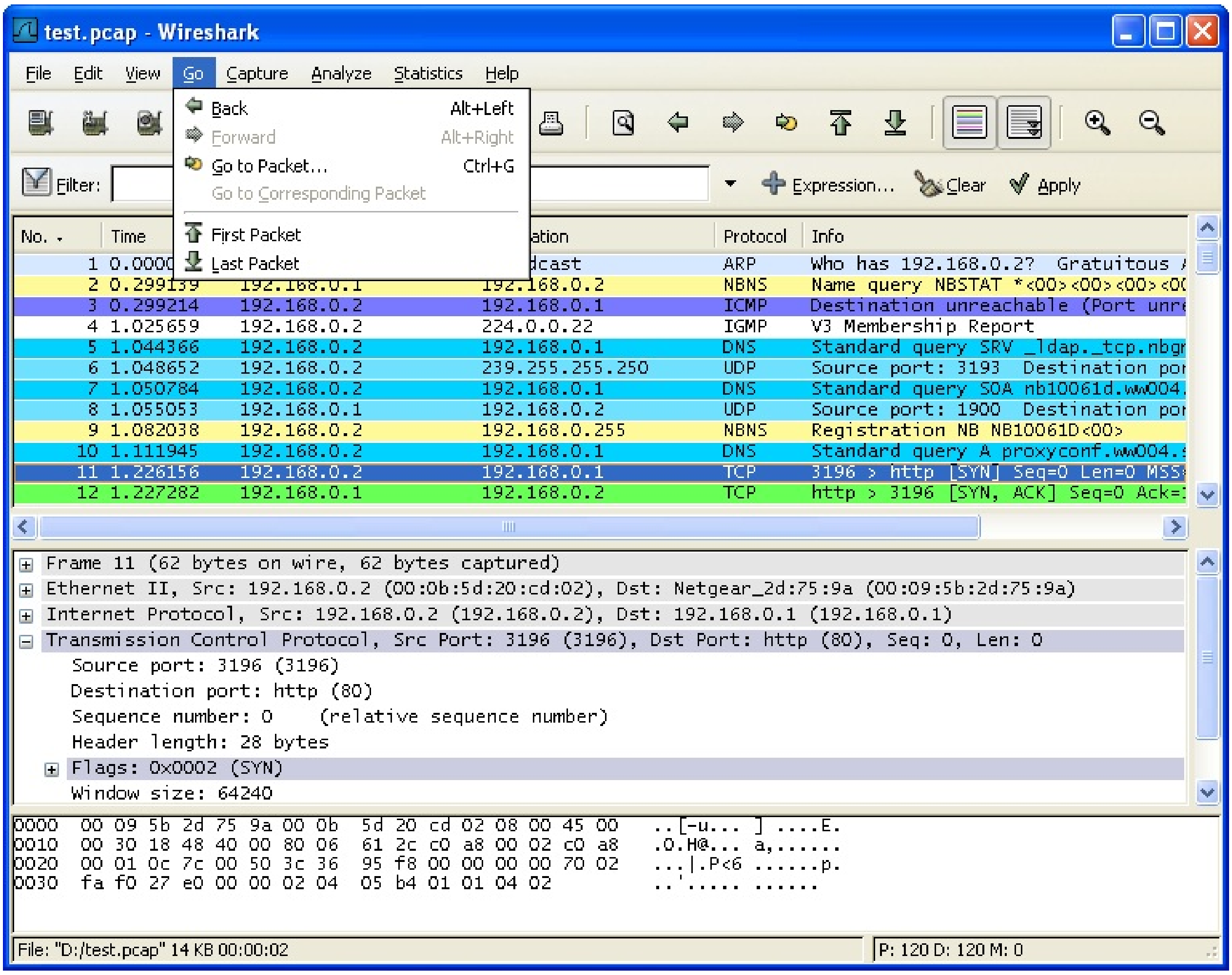


表 **3.5. "GO"菜单项**

菜单项	快捷键	描述
Back	Alt+Left	跳到最近浏览的包，类似于浏览器中的页面历史纪录
ForWard	Alt+Right	跳到下一个最近浏览的包，跟浏览器类似
Go to Packet	Ctrl+G	打开一个对话框，输入指定的包序号，然后跳转到对应的包，见第 6.8 节 “到指定的包”
Go to Corresponding Packet		跳转到当前包的应答包，如果不存在，该选项为灰色
Previous Packet	Ctrl+UP	移动到包列表中的前一个包，即使包列表面板不是当前焦点，也是可用的
Next Packet	Ctrl+Down	移动到包列表中的后一个包，同上
First Packet		移动到列表中的第一个包
Last Packet		移动到列表中的最后一个包

3.9. "Capture"菜单

"Capture"菜单的各项说明见表 3.6 “"Capture"菜单项”

图 **3.7. "Capture"菜单**

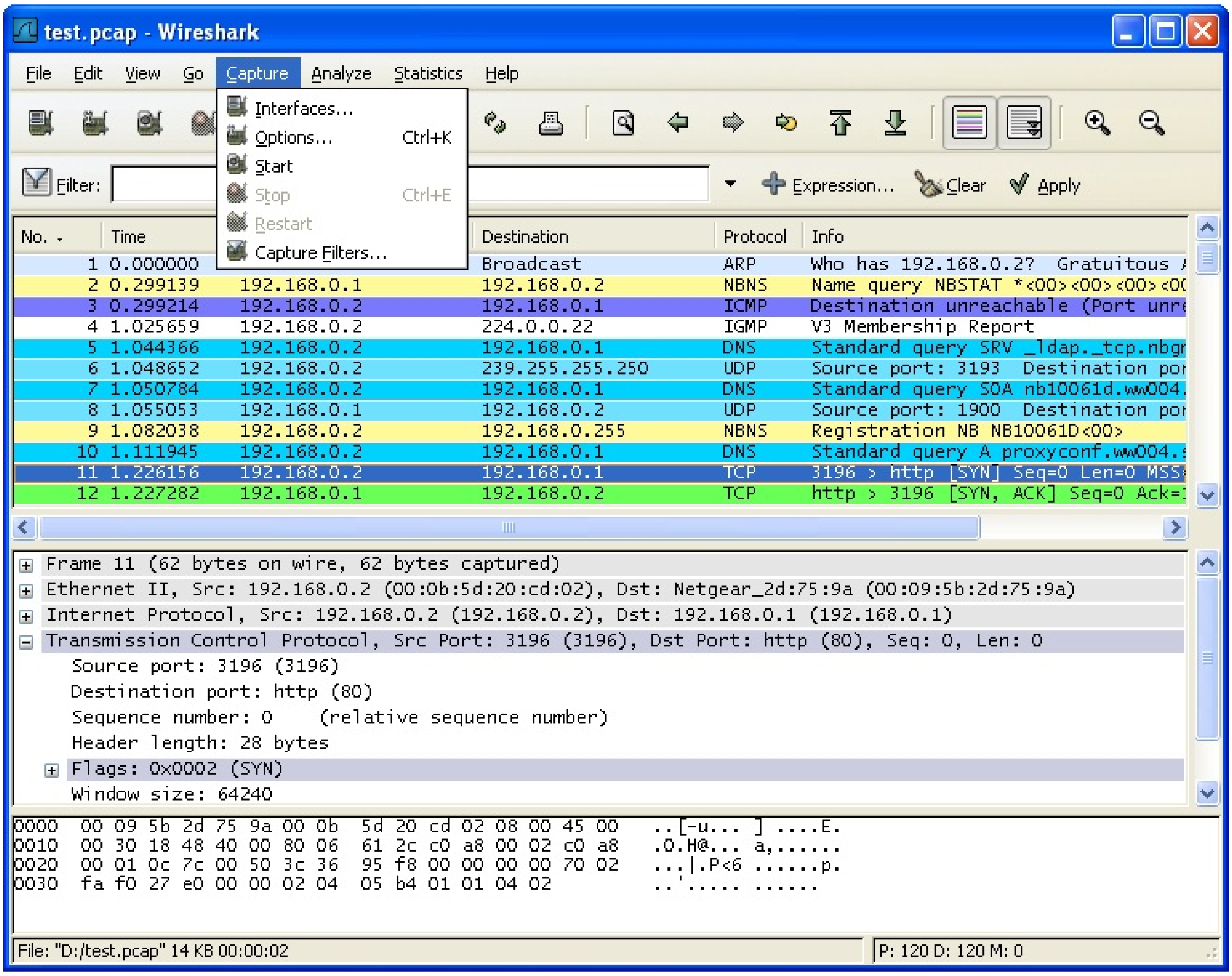


表 3.6. "Capture"菜单项

菜单项	快捷键	说明
Interface...		在弹出对话框选择您要进行捕捉的网络接口,见第 4.4 节 “捕捉接口对话框”
Options...	Ctrl+K	打开设置捕捉选项的对话框,(见第 4.5 节 “捕捉选项对话框”)并可以在此开始捕捉
Start		立即开始捕捉，设置都是参照最后一次设置。
Stop	Ctrl+E	停止正在进行的捕捉，见第 4.9.1 节 “停止捕捉”
Restart		正在进行捕捉时，停止捕捉，并按同样的设置重新开始捕捉.仅在您认为有必要时
Capture Filters...		打开对话框，编辑捕捉过滤设置，可以命名过滤器，保存为其他捕捉时使用见第 6.6 节 “定义，保存过滤器”

3.10. "Analyze"菜单

"Analyze"菜单的各项见表 3.7 “"analyze"菜单项”

图 **3.8. "Analyze"**菜单

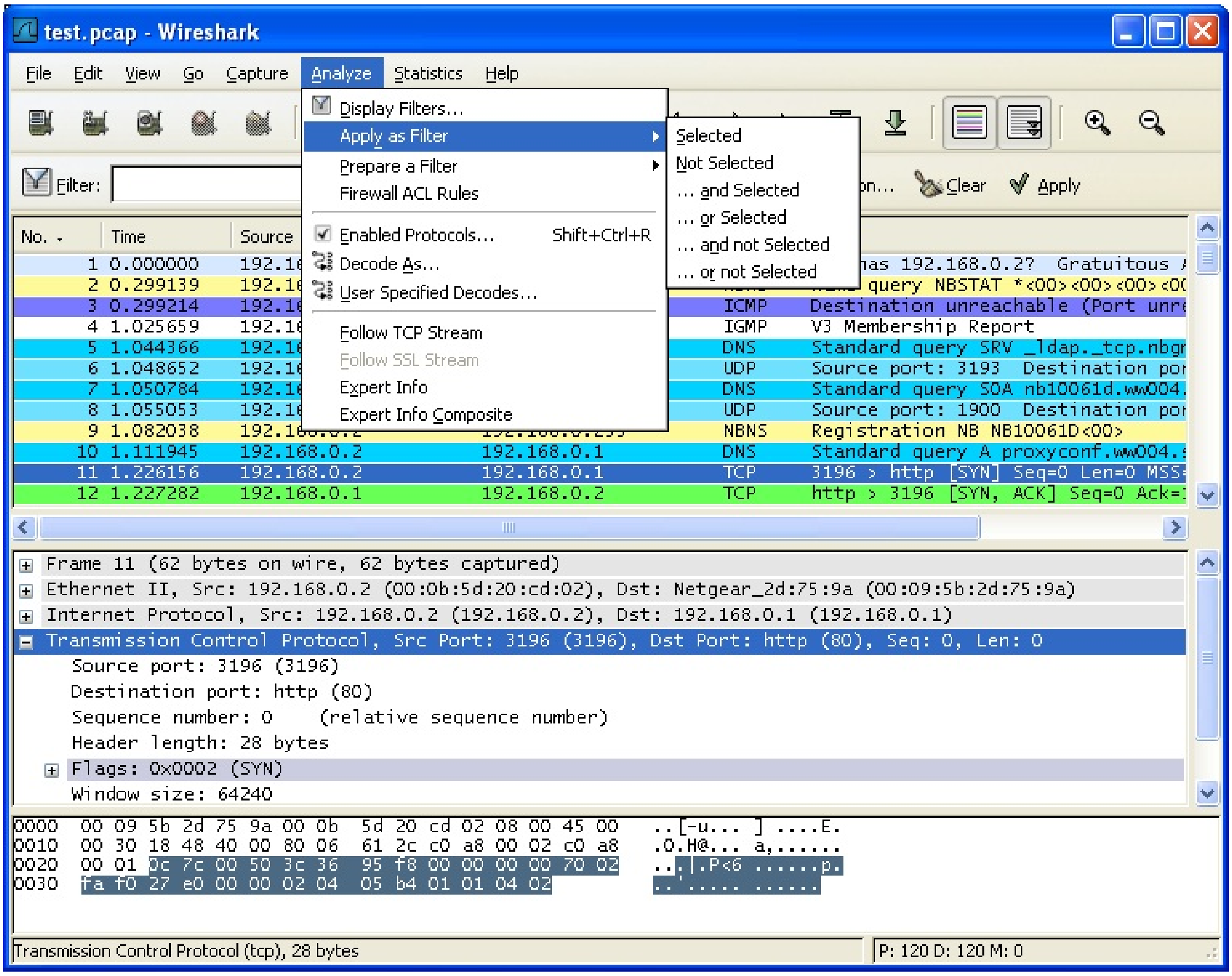


表 **3.7. "analyze"**菜单项

菜单项	快捷键	说明
Display Filters...		打开过滤器对话框编辑过滤设置，可以命名过滤设置，保存为其他地方使用，见第 6.6 节 "定义，保存过滤器"
Apply as Filter>...		更改当前过滤显示并立即应用。根据选择的项，当前显示字段会被替换成选择在Detail面板的协议字段
Prepare a Filter>...		更改当前显示过滤设置，当不会立即应用。同样根据当前选择项，过滤字符会被替换成Detail面板选择的协议字段
Firewall ACL Rules		为多种不同的防火墙创建命令行ACL规则(访问控制列表),支持Cisco IOS, Linux Netfilter (iptables), OpenBSD pf and Windows Firewall (via netsh). Rules for MAC addresses, IPv4 addresses, TCP and UDP ports, 以及IPv4+混合端口。以上假定规则用于外部接口。
Enable Protocols...	Shift+Ctrl+R	是否允许协议分析，见第 9.4.1 节 ""Enable Protocols"对话框"
		[a] 看样子他们有个关于这部分的章节

3.11. "Statistics"菜单

Wireshark "statistics"菜单项见表 3.8 “”

图 3.9. "Statistics"菜单

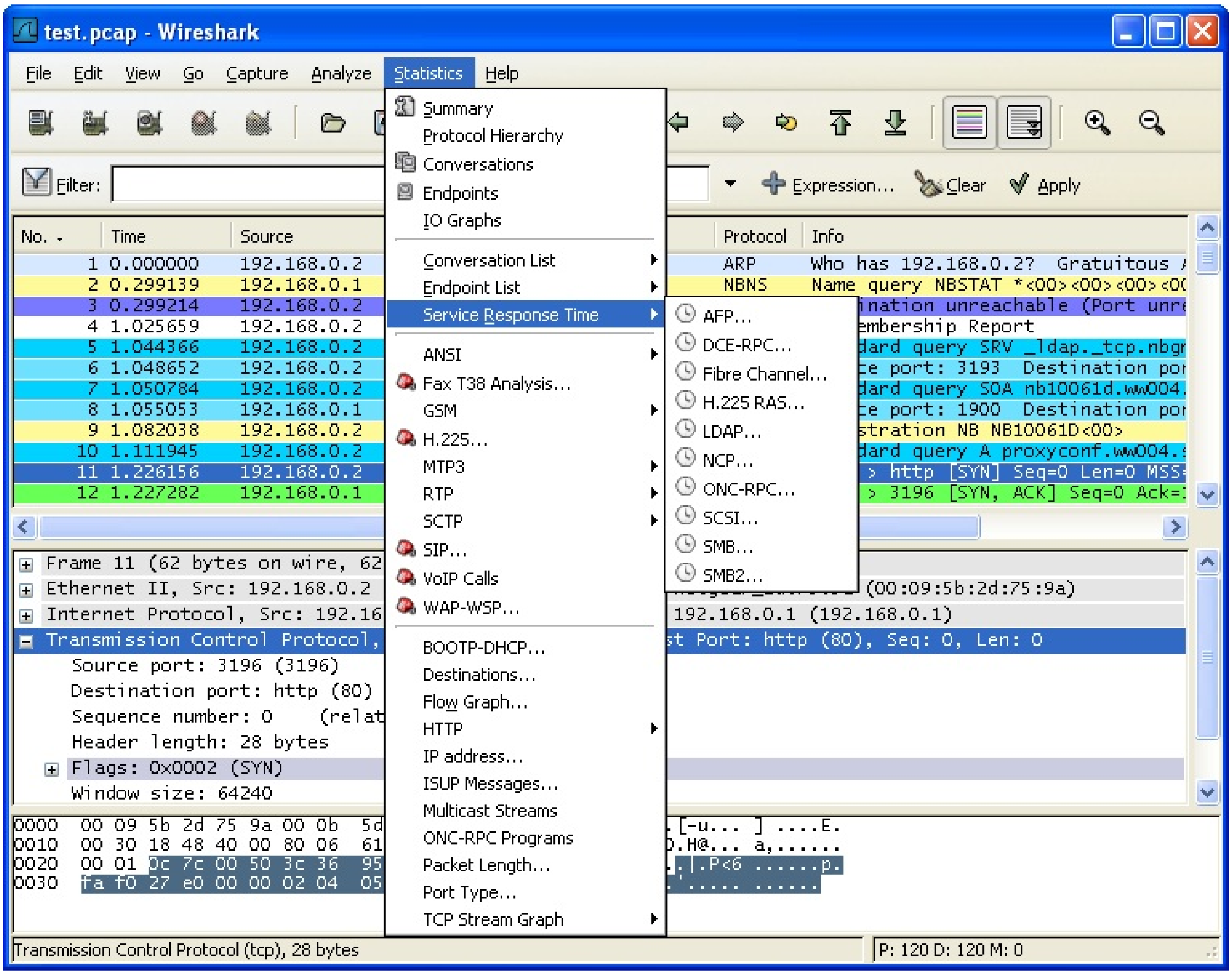


表 3.8.

菜单项	描述
Summary	显示捕捉数据摘要,见第 8.2 节 “摘要窗口”
Protocol Hierarchy	显示协议统计分层信息， 见第 8.3 节 “"Protocol Hierarchy"窗口”
Conversations/	显示会话列表(两个终端之间的通信),见???
EndPoints	显示端点列表(通信发起， 结束地址),见第 8.4.2 节 “"Endpoints"窗口”
IO Graphs	显示用户指定图表，(如包数量-时间表)见第 8.6 节 “"IO Graphs"窗口”
Conversation List	通过一个组合窗口， 显示会话列表,见第 8.5.3 节 “协议指定“Conversation List/会话列表”窗口”
Endpoint List	通过一个组合窗口显示终端列表， 见第 8.4.3 节 “特定协议的"Endpoint List"窗口”
Service Response Time	显示一个请求及其相应之间的间隔时间， 见第 8.7 节 “服务相应时间”
ANSI	见第 8.8 节 “协议指定统计窗口”
GSM	见第 8.8 节 “协议指定统计窗口”
H.225...	见第 8.8 节 “协议指定统计窗口”
ISUP Message	见第 8.8 节 “协议指定统计窗口”
Types	见第 8.8 节 “协议指定统计窗口”
MTP3	见第 8.8 节 “协议指定统计窗口”
RTP	见第 8.8 节 “协议指定统计窗口”
GSM	见第 8.8 节 “协议指定统计窗口”
SIP	见第 8.8 节 “协议指定统计窗口”
VOIP Calls...	见第 8.8 节 “协议指定统计窗口”
WAP-WSP...	见第 8.8 节 “协议指定统计窗口”
HTTP	HTTP请求/相应统计， 见第 8.8 节 “协议指定统计窗口”
ISUP Messages	见第 8.8 节 “协议指定统计窗口”
ONC-RPC Programs	见第 8.8 节 “协议指定统计窗口”
TCP Stream Graph	见第 8.8 节 “协议指定统计窗口”

3.12. "Help"菜单

帮助菜单的内容见表 3.9 “”

图 **3.10.** 帮助菜单

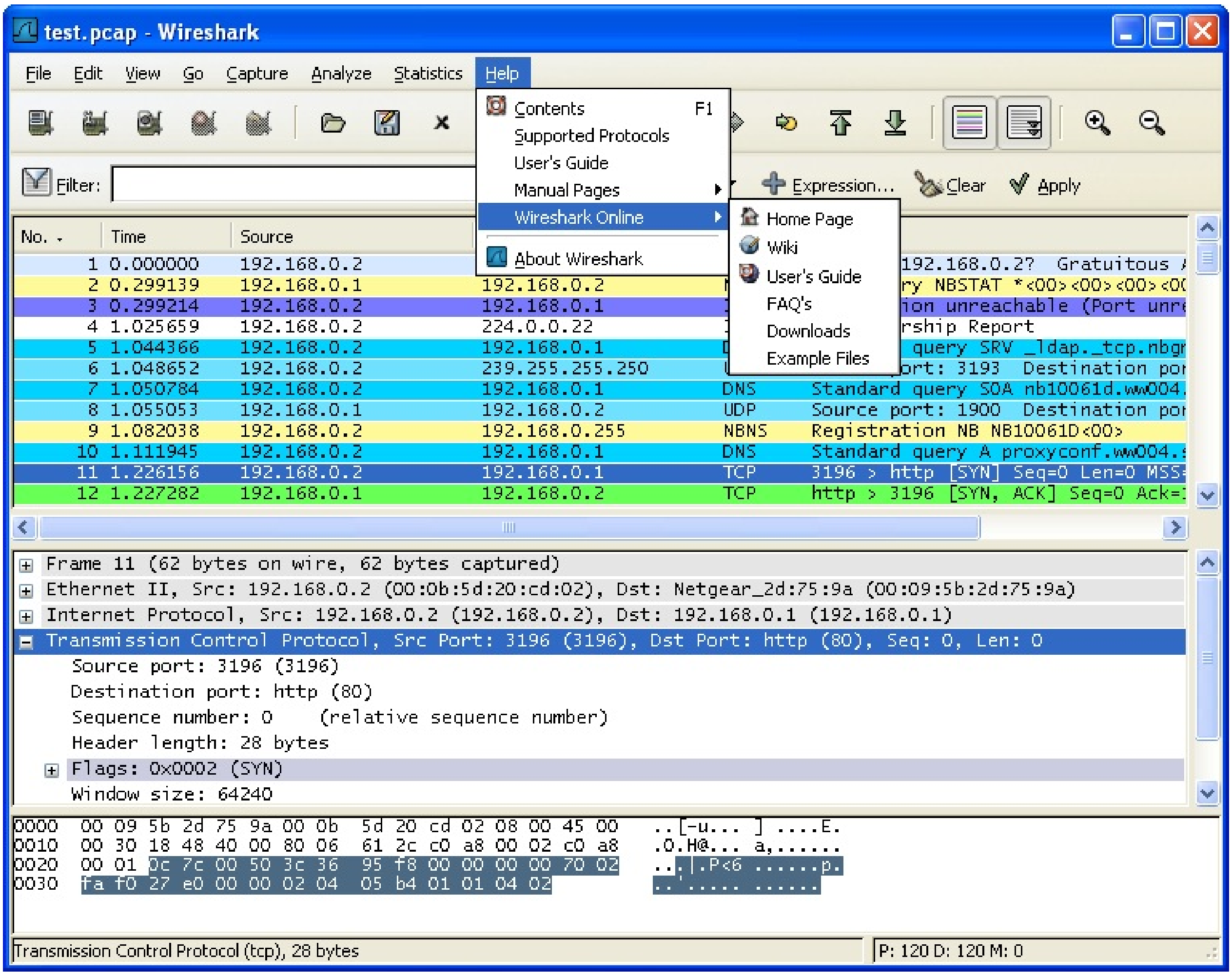


表 **3.9.**

菜单项	快捷键	描述
Contents	F1	打开一个基本的帮助系统
Supported Protocols		打开一个对话框显示支持的协议或工具
Manaul Pages>...		打开浏览器，显示安装在本地的手册
Wireshark Online>		按照选择显示在线资源
About Wireshark		弹出信息窗口显示Wireshark的一些相关信息，如插件，目录等。

注意

有些版本可能不支持调用WEB浏览器。如果是这样，可能会隐藏此菜单。

注意

如果调用浏览器错误，检查Wireshark首选项关于浏览器设置。

3.13. "Main"工具栏

主工具栏提供了快速访问常见项目的功能,它是不可以自定义的，但如果您觉得屏幕屏幕过于狭小，需要更多空间来显示数据。您可以使用浏览菜单隐藏它。

在主工具栏里面的项目只有在可以使用的时候才能被选择，如果不是可用则显示为灰色，不可选(例如：在未载入文件时，保存文件按钮就不可用.)

图 **3.11.**



表 **3.10.** 主工具栏选项

工具栏图标	工具栏项	对应菜单项	描述
	接口	Capture/Interfaces...	打开接口列表对话框,见第 4.3 节 “开始捕捉”
	选项。。。	Capture/Options	打开捕捉选项对话框，见第 4.4 节 “捕捉接口对话框”
	Start	Capture/Start	使用最后一次的捕捉设置立即开始捕捉
	STOP	Capture/Stop	停止当前的捕捉，见第 4.3 节 “开始捕捉”
	Restar	Caputer/Rstart	停止当前捕捉，并立即重新开始
	Open...	File/Open	启动打开文件对话框，用于载入文件，详见第 5.2.1 节 “打开捕捉文件对话框”
	Save As...	File/Save As...	保存当前文件为任意其他的文件，它将会弹出一个对话框，(见第 5.3.1 节 “"save Capture File As/保存文件为"对

			话框”
			注意
			如果当前文件是临时未保存文件，图标将会显示为
	Close	File/Close	关闭当前文件。如果未保存，将会提示是否保存
	Reload	View/Reload	重新载入当前文件
	Print	File/Print	打印捕捉文件的全部或部分，将会弹出一个打印对话框(见第 5.7 节 “打印包”)
	Find packet...	Edit/Find Packet...	打开一个对话框，查找包。见第 6.7 节 “查找包”
	Go Back	Go/Go Back	返回历史记录中的上一个
	Go Forward	Go/Go Forward	跳转到历史记录中的下一个包
	Go to Packet...	Go/Go to Packet...	弹出一个设置跳转到指定的包的对话框
	Go To First Packet	Go/First Packet	跳转到第一包
	Go To Last Packet	Go/Last Packet	跳转到最后一个包
	Colorize	View/Coloreze	切换是否以彩色方式显示包列表
	Auto Scroll in Live	View/Auto Scrool in Live Capture	开启/关闭实时捕捉时自动滚动包列表
	Zoom in	View/Zoom In	增大字体
	zoom out	View/Zoom Out	缩小字体
	Normal Size	View/Normal Size	设置缩放大小为100%
	Resize Columns	View/Resize Columns	重置列宽，使内容适合列宽(使包列表内的文字可以完全显示)
	Capture Filters..	Capture/Capture Filters...	打开对话框，用于创建、编辑过滤器。详见第 6.6 节 “定义，保存过滤器”
	Display Filters..	Analyze/ Filters...	打开对话框，用于创建、编辑过滤器。详见第 6.6 节 “定义，保存过滤器”
	Coloring Rules...	View/Coloring Rules...	定义以色彩方式显示数据包的规则详见第 9.3 节 “包色彩显示设置”
			打开首选项对话框，详见第 9.5 节 “首

			选项”
	Help	Help/Contents	打开帮助对话框

3.14. "Filter"工具栏

过滤工具栏用于编辑或显示过滤器，更多详情见第 6.3 节 “浏览时过滤包”

图 **3.12.** 过滤工具栏



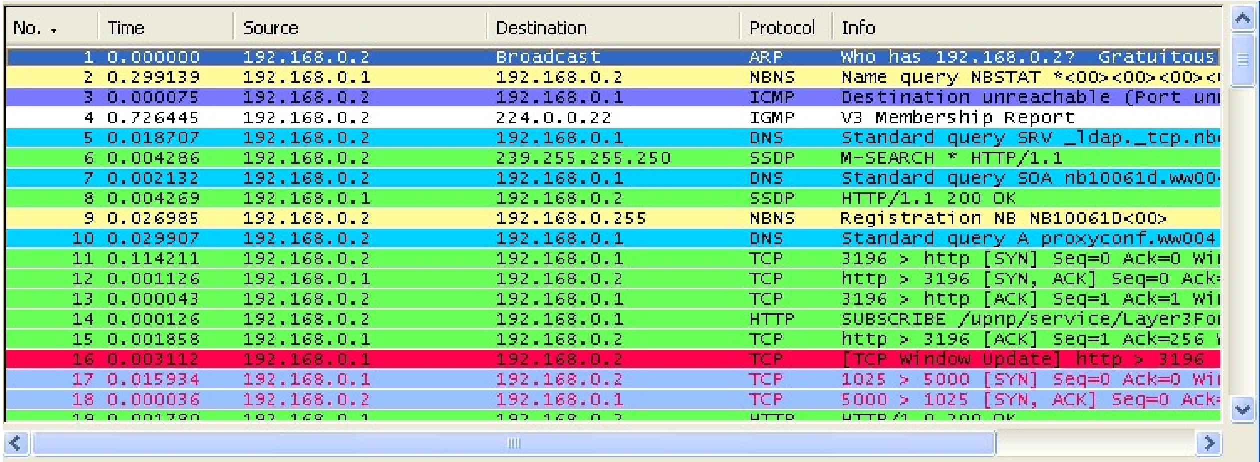
表 **3.11.**

工具栏图标	工具栏项	说明
	过滤	打开构建过滤器对话框,见第 6.7 节 “查找包”[a]
	过滤输入框	在此区域输入或修改显示的过滤字符，见第 6.4 节 “建立显示过滤表达式”，在输入过程中会进行语法检查。如果您输入的格式不正确，或者未输入完成，则背景显示为红色。直到您输入合法的表达式，背景会变为绿色。你可以点击下拉列表选择您先前键入的过滤字符。列表会一直保留，即使您重新启动程序。
		注意
		做完修改之后，记得点击右边的Apply(应用)按钮，或者回车，以使过滤生效。
		注意
		输入框的内容同时也是当前过滤器的内容（当前过滤器的内容会反映在输入框）
	表达式...	标签为表达式的按钮打开一个对话框用以从协议字段列表中编辑过滤器，详见第 6.5 节 ““Filter Expression/过滤表达式”对话框”
	清除	重置当前过滤器，清除输入框
	应用	应用当前输入框的表达式为过滤器进行过滤
		注意
		在大文件里应用显示过滤可能要很长时间
		[a] 我看到的Filter按钮貌似没有图标，可能只出现在0.99.4版中

3.15. "Pcaket List"面板

Packet list/包列表面板显示所有当前捕捉的包

图 3.13. **"Packet list/包列表"面板**



列表中的每行显示捕捉文件的一个包。如果您选择其中一行，该包得更多情况会显示在"Packet Detail/包详情"，"Packet Byte/包字节"面板

在分析(解剖)包时，Wireshark会将协议信息放到各个列。因为高层协议通常会覆盖底层协议，您通常在包列表面板看到的都是每个包的最高层协议描述。

例如：让我们看看一个包括TCP包,IP包,和一个以太网包。在以太网(链路层？)包中解析的数据(比如以太网地址），在IP分析中会覆盖为它自己的内容(比如IP地址)，在TCP分析中会覆盖IP信息。

包列表面板有很多列可供选择。需要显示哪些列可以在首选项中进行设置，见第 9.5 节 “首选项”

默认的列如下

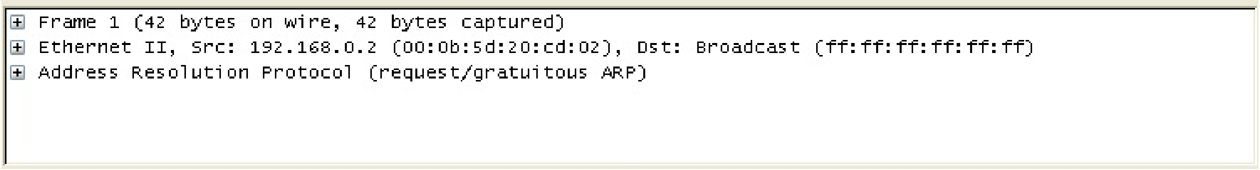
- **No.** 包的编号，编号不会发生改变，即使进行了过滤也同样如此
- **Time** 包的时间戳。包时间戳的格式可以自行设置，见第 6.10 节 “时间显示格式及参考时间”
- **Source** 显示包的源地址。
- **Destination** 显示包的目标地址。
- **Protocal** 显示包的协议类型的简写
- **Info** 包内容的附加信息

右击包，可以显示对包进行相关操作的上下文菜单。见第 6.3 节 “浏览时过滤包”

3.16. "Packet Details"面板

"Packet Details/包详情"面板显示当前包(在包列表面板被选中的包)的详情列表。

图 3.14. **"Packet Details/包详情"面板**



该面板显示包列表面板选中包的协议及协议字段，协议及字段以树状方式组织。你可以展开或折叠它们。

右击它们会获得相关的上下文菜单。见第 6.4 节 “建立显示过滤表达式”

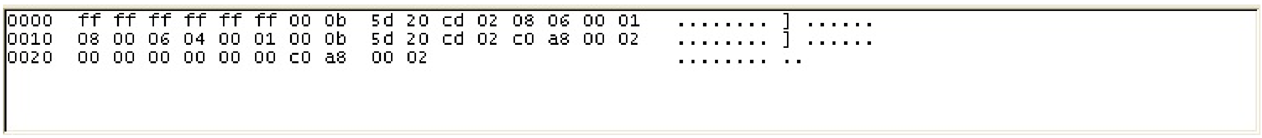
某些协议字段会以特殊方式显示

- **Generated fields/**衍生字段 Wireshark会将自己生成附加协议字段加上括号。衍生字段是通过该包的相关的其他包结合生成的。例如：Wireshark 在对TCP流应答序列进行分析时。将会在TCP协议中添加[SEQ/ACK analysis]字段
- **Links/**链接 如果Wireshark检测到当前包与其它包的关系，将会产生一个到其它包的链接。链接字段显示为蓝色字体，并加有下划线。双击它会跳转到对应的包。

3.17. "Packet Byte"面板

Packet Byte/包字节 面板以16进制转储方式显示当前选择包的数据

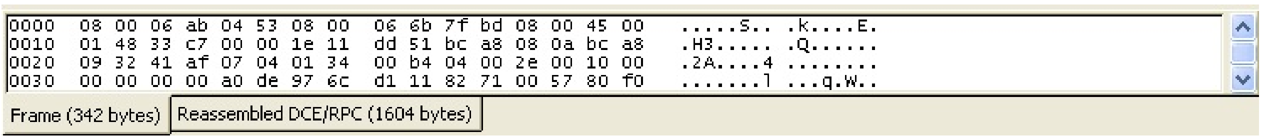
图 3.15. **Packet Byte/**包字节面板



通常在16进制转储形式中，左侧显示包数据偏移量，中间栏以16进制表示，右侧显示为对应的ASCII字符

根据包数据的不同，有时候包字节面板可能会有多个页面，例如：有时候Wireshark会将多个分片重组为一个，见第 7.5 节 “重组包”.这时会在面板底部出现一个附加按钮供你选择查看

图 **3.16.** 带选项的**"Paket Bytes/**包字节**"**面板



注意

附加页面的内容可能来自多个包。

右击选项按钮会显示一个上下文菜单显示所有可用的页的清单。如果您的面板尺寸过小，这项功能或许有所帮助

3.18. 状态栏

状态栏用于显示信息

通常状态栏的左侧会显示相关上下文信息，右侧会显示当前包数目

图 **3.17.** 初始状态栏



该状态栏显示的是没有文件载入时的状态，如：刚启动Wireshark时

图 **3.18.** 载入文件后的状态栏



左侧显示当前捕捉文件信息，包括名称，大小，捕捉持续时间等。

右侧显示当前包在文件中的数量，会显示如下值

- P:捕捉包的数目
- D:被显示的包的数目
- M: 被标记的包的数目.

图 **3.19.** 已选择协议字段的状态栏



如果你已经在"Packet Detail/包详情"面板选择了一个协议字段，将会显示上图

> 提示
>
> 括号内的值(如上图的app.opcode)可以作为显示过滤使用。它表示选择的协议字段。

第 4 章 实时捕捉数据包

目录

4.1. 介绍

实时捕捉数据包时Wireshar的特色之一

Wiershark捕捉引擎具备以下特点

- 支持多种网络接口的捕捉(以太网，令牌环网，ATM...)
- 支持多种机制触发停止捕捉，例如：捕捉文件的大小，捕捉持续时间，捕捉到包的数量...
- 捕捉时同时显示包解码详情
- 设置过滤，减少捕捉到包的容量。见第 4.8 节 “捕捉时过滤”
- 长时间捕捉时，可以设置生成多个文件。对于特别长时间的捕捉，可以设置捕捉文件大小罚值，设置仅保留最后的N个文件等手段。见第 4.6 节 “捕捉文件格式、模式设置”

Wireshark捕捉引擎在以下几个方面尚有不足

- 从多个网络接口同时实时捕捉，(但是您可以开始多个应用程序实体，捕捉后进行文件合并)
- 根据捕捉到的数据停止捕捉(或其他操作)

4.2. 准备工作

第一次设置Wireshark捕捉包可能会遇到一些小麻烦

> 提示
>
> 关于如何进行捕捉设置的较为全面的向导可以在:http://wiki.wireshark.org/CaptureSetup.

这里有一些常见需要注意的地方

- 你必须拥有root/Administrator特权以开始捕捉[12]
- 必须选择正确的网络接口捕捉数据
- 如果您想捕捉某处的通信，你必须作出决定：在什么地方可以捕捉到
- ……以及许多

如果你碰到设置问题，建议看看前面的那个向导，或许会有所帮助

[12] 记得在Windows安装那一节层提到如果作为服务启动可以避免非管理员无法进行捕捉，不知道二者能否相互印证。

4.3. 开始捕捉

可以使用下任一方式开始捕捉包

- 使用打开捕捉接口对话框，浏览可用的本地网络接口，见图 4.1 “"Capture Interfaces"捕捉接口对话框”,

  选择您需要进行捕捉的接口启动捕捉
- 你也可以使用"捕捉选项"按钮启动对话框开始捕捉，见图 4.2 “"Capture Option/捕捉选项"对话框”
- 如果您前次捕捉时的设置和现在的要求一样，您可以点击"开始捕捉"按钮或者是菜单项立即开始本次捕捉。
- 如果你已经知道捕捉接口的名称，可以使用如下命令从命令行开始捕捉：

```
wireshark -i eth0 -k
```

上述命令会从eht0接口开始捕捉，有关命令行的介绍参见第 9.2 节 “从命令行启动 Wireshark”

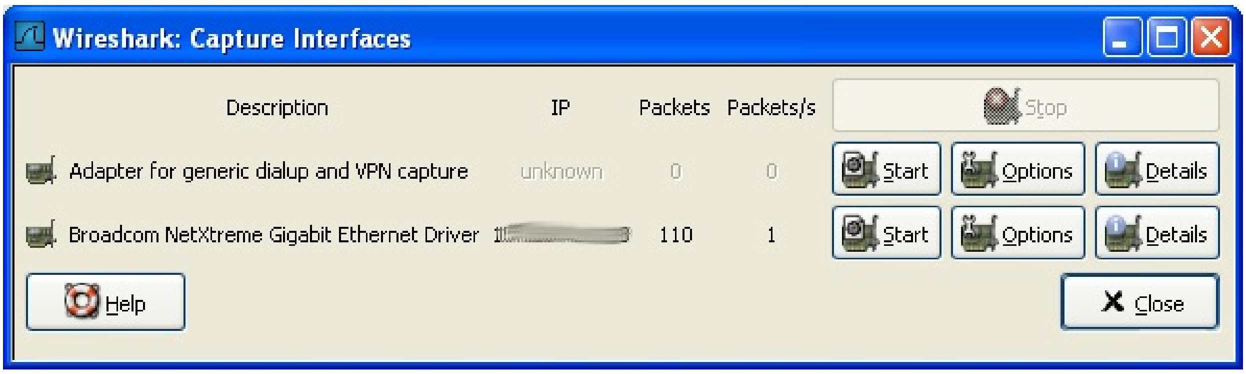
4.4. 捕捉接口对话框

如果您从捕捉菜单选择"Interface...", 将会弹出如图 4.1 “"Capture Interfaces"捕捉接口对话框”所示的对话框

> 警告
>
> 打开"Capture Interfaces"/捕捉对话框时 同时正在显示捕捉的数据，这将会大量消耗您的系统资源。尽快选择您需要的接口以结束该对话框。避免影响系统性能

> 注意
>
> 这个对话框只显示本地已知的网络接口，Wireshark可能无法检测到所有的本地接口，Wireshark不能检测远程可用的网络接口，Wireshark只能使用列出可用的网络接口

图 4.1. "Capture Interfaces"捕捉接口对话框



描述

从操作系统获取的接口信息

IP

Wireshark能解析的第一个IP地址，如果接口未获得IP地址（如，不存在可用的DHCP服务器)，将会显示"Unkow",如果有超过一个IP的，只显示第一个(无法确定哪一个会显示).

Packets

打开该窗口后，从此接口捕捉到的包的数目。如果一直没有接收到包，则会显示为灰度

Packets/s

最近一秒捕捉到包的数目。如果最近一秒没有捕捉到包，将会是灰度显示

Stop

停止当前运行的捕捉

Capture

从选择的接口立即开始捕捉，使用最后一次捕捉的设置。

Options

打开该接口的捕捉选项对话框，见 第 4.5 节 “捕捉选项对话框”

Details(仅Win32系统)

打开对话框显示接口的详细信息

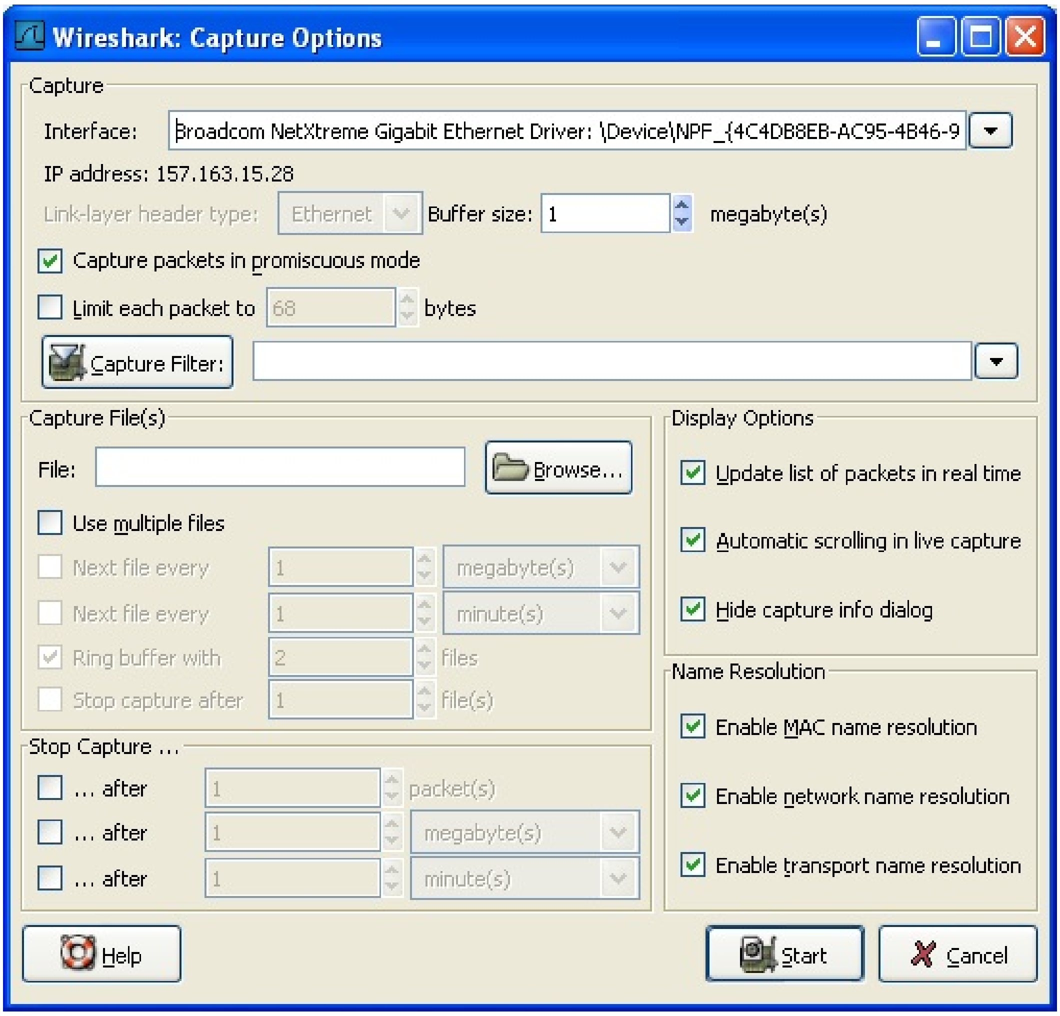
Close

关闭对话框

4.5. 捕捉选项对话框

如果您从捕捉菜单选择"start..."按钮(或者从主工具栏选择对应的项目),Wireshark弹出"Capture Option/捕捉选项"对话框。如图 4.2 “"Capture Option/捕捉选项"对话框”所示

图 **4.2. "Capture Option/捕捉选项"**对话框



提示

如果你不了解各项设置的意义，建议保持默认。

你可以用对话框中的如下字段进行设置

4.5.1. 捕捉桢

Interface

该字段指定你想用于进行捕捉的借口。一次只能使用一个接口。这是一个下拉列表，简单点击右侧的按钮，选择你想要使用的接口。默认第一是支持捕捉的non-loopback(非环回)接口，如果没有这样的接口，第一个将是环回接口。在某些系统中，回借口不支持捕捉包(windows平台下的环回接口就不支持。)

在命令行使用-i <interface>参数可以替代该选项

IP address

表示选择接口的IP地址。如果系统未指定IP地址，将会显示为"unknown"

Link-layer header type

除非你有些特殊应用，尽量保持此选项默认。想了解更多详情，见 第 4.7 节 “链路层包头类型”

Buffer size: n megabyte(s)

输入用于捕捉的缓层大小。该选项是设置写入数据到磁盘前保留在核心缓存中捕捉数据的大小，如果你发现丢包。尝试增大该值。

> 注意
>
> 该选项仅适用于Windows平台

Capture packets in promiscuous mode

指定Wireshark捕捉包时，设置接口为杂收模式(有些人翻译为混杂模式)。如果你未指定该选项，Wireshark 将只能捕捉进出你电脑的数据包(不能捕捉整个局域网段的包)[13]

> 注意
>
> 如果其他应用程序将网卡设置为杂收模式，即使不选中该选项，也会工作于杂收模式下。
>
> 注意
>
> 即使在杂收模式下，你也未必能够接收到整个网段所有的网络包。详细解释见 http://www.wireshark.org/faq.html#promiscsniff

Limit each packet to n bytes

指定捕捉过程中，每个包的最大字节数。在某些地方被称为。"snaplen".[14]如果禁止该选项，默认值为65535，这适用于大多数协议，下面是一些大多数情况下都适用的规则(这里又出现了拇指规则，第一章，系统需求时提到过。这里权且翻译作普适而非绝对的规则))

- 如果你不确定，尽量保持默认值
- 如果你不需要包中的所有数据。例如：如果您仅需要链路层、IP和TCP包头，您可能想要选择一个较小的快照长度。这样只需要较少的cpu占用时间用于复制包，包需要的缓存也较少。如此在繁忙网络中捕捉时丢失的包也可能会相应少一点。

- 如果你没有捕捉包中的所有数据(适用snpaplen截断了包)，你可能会发现有时候你想要的包中的数据部分被截断丢弃了。或者因为缺少重要的部分，想对某些包进行重组而发现失败。

Capture Filter

指定捕捉过滤。捕捉过滤器将会在有第 4.8 节 “捕捉时过滤”详细介绍，默认情况下是空的。

同样你也可以点击捕捉按钮，通过弹出的捕捉过滤对话框创建或选择一个过滤器，详见第 6.6 节 “定义，保存过滤器”

4.5.2. 捉数据帧为文件。

捕捉文件设置的使用方法的详细介绍见第 4.6 节 “捕捉文件格式、模式设置”

File

指定将用于捕捉的文件名。该字段默认是空白。如果保持空白，捕捉数据将会存储在临时文件夹。详见第 4.6 节 “捕捉文件格式、模式设置”

你可以点击右侧的按钮打开浏览窗口设置文件存储位置

Use multiple files

如果指定条件达到临界值，Wireshark将会自动生成一个新文件，而不是适用单独文件。

Next file every n megabyte(s)

仅适用选中Use multiple files,如果捕捉文件容量达到指定值，将会生成切换到新文件

Next file every n minutes(s)

仅适用选中Use multiple files,如果捕捉文件持续时间达到指定值，将会切换到新文件。

Ring buffer with n files

仅适用选中Use multiple files,仅生成制定数目的文件。

Stop caputure after n file(s)

仅适用选中Use multiple files,当生成指定数目文件时，在生成下一个文件时停止捕捉(生成n个还是n+1个文件?)

4.5.3. 停止捕捉桢

... after n packet(s)

在捕捉到指定数目数据包后停止捕捉

... after n megabytes(s)

在捕捉到指定容量的数据(byte(s)/kilobyte(s)/megabyte(s)/gigabyte(s))后停止捕捉。如果没有适用"user multiple files",该选项将是灰色

... after n minute(s)

在达到指定时间后停止捕捉

4.5.4. 显示桢选项

Update list of packets in real time

在包列表面板实时更新捕捉数据。如果未选定该选项，在Wireshark捕捉结束之前将不能显示数据。如果选中该选项，Wireshark将生成两个独立的进程，通过捕捉进程传输数据给显示进程。

Automatic scrolling in live capture

指定Wireshark在有数据进入时实时滚动包列表面板，这样您将一直能看到最近的包。反之，则最新数据包会被放置在行末，但不会自动滚动面板。如果未设置"update list of packets in real time",该选项将是灰色不可选的。

Hide capture info dialog

选中该选项，将会隐藏捕捉信息对话框

4.5.5. 名称解析设置

Enable MAC name resolution

设置是否让Wireshark翻译MAC地址为名称，见第 7.6 节 “名称解析”

Enable network name resolution

是否允许Wireshark对网络地址进行解析，见第 7.6 节 “名称解析”

4.5.6. 按钮

进行完上述设置以后，你可以点击**start**按钮进行捕捉,也可以点击**Cancel**退出捕捉.

开始捕捉以后，在你收集到足够的数据时你可以停止捕捉。见第 4.9 节 “在捕捉过程中”

[13] 网卡在局域网内会接到很多不属于自己的包，默认情况下，网卡会不对这些包进行处理。貌似设置为杂收模式，Wireshak会监听所有的包，但并不作出相应。

[14] 粗略查了一下,未找到该词的合适翻译,多见于Winpcap的描述，如果把该单词拆分，snap:单元，快照，len:长度，似乎就是单位长度，单元大小的意思。在看看该段下面第二个如果中提到的snapshot length,snaplen应该是二者的简写形式，快照长度

4.6. 捕捉文件格式、模式设置

在 捕捉时，libpcap 捕捉引擎(linux环境下)会抓取来自网卡的包存放在(相对来说)较小的核心缓存内。这些数据由Wireshark读取并保存到用户指定的捕捉文件中。

保存包数据到捕捉文件时，可采用差异模式操作。

> 提示
>
> 处理大文件(数百兆)将会变得非常慢。如果你计划进行长时间捕捉，或者处于一个高吞吐量的网络中，考虑使用前面提到的"Multiple files/多文件"选项。该选项可以将捕捉包分割为多个小文件。这样可能更适合上述环境。
>
> 注意
>
> 使用多文件可能会切断上下文关联信息。Wireshark保留载入包的上下文信息，所以它会报告上下文关联问题(例如流问题)和关联上下文协议信息(例如：何处数据产生建立阶段，必须查找后续包)。这些信息仅能在载入文件中显示，使用多文件模式可能会截断这样的上下文。如果建立连接阶段已经保存在一个文件中，你想要看的在另一个文件中，你可能无法看到可用的上下文关联信息。
>
> 提示
>
> 关于捕捉文件的目录信息，可见???

表 **4.1.** 捕捉文件模式选项

"File"选项	"Use multiple files"选项	"Ring buffer with n files"选项	Mode	最终文件命名方式
-	-	-	Single temporary file	etherXXXXXX (where XXXXXX 是一个独立值)
foo.cap	-	-	Single named file	foo.cap
foo.cap	x	-	Multiple files,continuous	foo_00001_20040205110102.cap, foo_00002_20040205110102.cap, ...
foo.cap	x	x	Multiple files,ring buffer	foo_00001_20040205110102.cap, foo_00002_20040205110102.cap, ...

Single temporary file

将会创建并使用一个临时文件(默认选项).捕捉文件结束后，该文件可以由用户指定文件名。

Single named file

使用单独文件，如果你想放到指定目录，选择此模式

Multiple files,continuous

与single name file模式类似，不同点在于，当捕捉达到多文件切换临界条件时之一时，会创建一个新文件用于捕捉

Multiple files,ring buffer

与"multiple files continuous"模式类似，不同之处在于，创建的文件数目固定。当达到ring buffer with n值时，会替换掉第一个文件开始捕捉，如此循环往复。

该模式可以限制最大磁盘空间使用量，即使未限制捕捉数据输入，也只能保留最后几个捕捉数据。

4.7. 链路层包头类型

在通常情况下，你不需要选择链路层包头类型。下面的段落描述了例外的情况，此时选择包头类型是有必要的，所以你需要知道怎么做：

如果你在某种版本BSD操作系统下从某种802.11 设备(无线局域网设备)捕捉数据，可能需要在"802.11"和"Ethernet"中做出选择。"Ethernet"将会导致捕捉到的包带有伪以太网帧头(不知道是不是应该叫伪首部更准确些);"802.11"将会导致他们带有802.11帧头。如果捕捉时的应用程序不支持"802.11帧头"，你需要选择"802.11"

如果你使用Endace DAG card(某种网络监视卡)连接到同步串口线(译者注：E文为synchronous serial line，权且翻译作前文吧，未接触过此卡、未熟稔此线名称)，可能会出现"PPP over serial" 或 "Cisco HDLC"(自己google去)供选择。根据你自己的情况选择二者中的一个。

如果你使用Endace DAG card(同上)连接到ATM网络，将会提供"RFC 1483 IP-over-ATM"、"Sun raw ATM"供选择。如果捕捉的通信是RFC 1483封装IP(RFC 1483 LLC-encapsulated IP,不翻译为妙)，或者需要在不支持SunATM帧头的应用程序下捕捉，选择前者。反之选择后者。

如果你在以太网捕捉，将会提供"Ethernet"、"DOCSIS"供选择，如果您是在Cisco Cable Modem Termination System(CMTS是思科同轴电缆终端调制解调系统？)下捕捉数据。它会将DOCSIS(同轴电缆数据服务接口)通信放置到以太网中，供捕捉。此时需要选择"DOCSIS",反之则反之。

4.8. 捕捉时过滤

Wireshark使用libpcap过滤语句进行捕捉过滤(what about winpcap?)。在tcpdump主页有介绍，但这些只是过于晦涩难懂，所以这里做小幅度讲解。

> 提示
>
> 你可以从http://wiki.wireshark.org/CaptureFilters找到捕捉过滤范例.

在Wireshark捕捉选项对话(见图 4.2 “"Capture Option/捕捉选项"对话框”)框输入捕捉过滤字段。下面的语句有点类似于tcpdump捕捉过滤语言。在tcpdump主页http://www.tcpdump.org/tcpdump_man.html可以看到tcpdump表达式选项介绍。

捕捉过滤的形式为：和取值(**and/or**)进行进行基本单元连接，加上可选的，高有限级的**not**:

```
[not] **primitive** [and|or [not] **primitive** ...]
```

例 **4.1.** 捕捉来自特定主机的**telnet**协议

```
tcp port 23 and host 10.0.0.5
```

本例捕捉来自或指向主机10.0.0.5的Telnet 通信，展示了如何用and连接两个基本单元。另外一个例子例 4.2 “捕捉所有不是来自10.0.0.5的telnet 通信”展示如何捕捉所有不是来自10.0.0.5的telnet 通信。

例 **4.2.** 捕捉所有不是来自**10.0.0.5**的**telnet** 通信

```
tcp host 23 and not src host 10.0.0.5
```

此处笔者建议增加更多范例。但是并没有添加。

一个基本单元通常是下面中的一个

[src|dst] host <host>

此基本单元允许你过滤主机ip地址或名称。你可以优先指定src|dst关键词来指定你关注的是源地址还是目标地址。如果未指定，则指定的地址出现在源地址或目标地址中的包会被抓取。

ether [src|dst] host <ehost>

此单元允许你过滤主机以太网地址。你可以优先指定关键词src|dst在关键词ether和host之间，来确定你关注的是源地址还是目标地址。如果未指定，同上。

gateway host<host>

过滤通过指定**host**作为网关的包。这就是指那些以太网源地址或目标地址是**host**，但源ip地址和目标ip地址都不是**host**的包

[src|dst] net <net> [{mask<mask>}|{len <len>}]

通过网络号进行过滤。你可以选择优先指定**src|dst**来确定你感兴趣的是源网络还是目标网络。如果两个都没指定。指定网络出现在源还是目标网络的都会被选择。另外，你可以选择子网掩码或者CIDR(无类别域形式)。

[tcp|udp] [src|dst] port <port>

过滤tcp,udp及端口号。可以使用**src|dst**和**tcp|udp**关键词来确定来自源还是目标，**tcp**协议还是**udp**协议。**tcp|udp**必须出现在**src|dst**之前。

less|greater <length>

选择长度符合要求的包。（大于等于或小于等于）

ip|ether proto <protocol>

选择有指定的协议在以太网层或是ip层的包

ether|ip broadcast|multicast

选择以太网/ip层的广播或多播

<expr> relop <expr>

创建一个复杂过滤表达式，来选择包的字节或字节范围符合要求的包。请参考http://www.tcpdump.org/tcpdump_man.html

4.8.1. 自动过滤远程通信

如果Wireshark是使用远程连接的主机运行的(例如使用SSH,X11 Window输出，终端服务器)，远程连接必须通过网络传输，会在你真正感兴趣的通信中产生大量数据包(通常也是不重要的)

想要避免这种情况，wireshark可以设置为如果发现有远程连接(通过察看指定的环境变量)，自动创建一个过滤器来匹配这种连接。以避免捕捉Wireshark捕捉远程连接通信。

下列环境变量可以进行分析

SSH——CONNECTION(ssh)

```
&lt;remote IP&gt; &lt;remote port&gt; &lt;local IP&gt; &lt;local port&gt;
```

SSH_CLIENT (ssh)

```
&lt;remote IP&gt; &lt;remote port&gt; &lt;local port&gt;
```

REMOTEHOST (tcsh, others?)

```
&lt;remote name&gt;
```

DISPLAY (x11)

```
[remote name]:&lt;display num&gt;
```

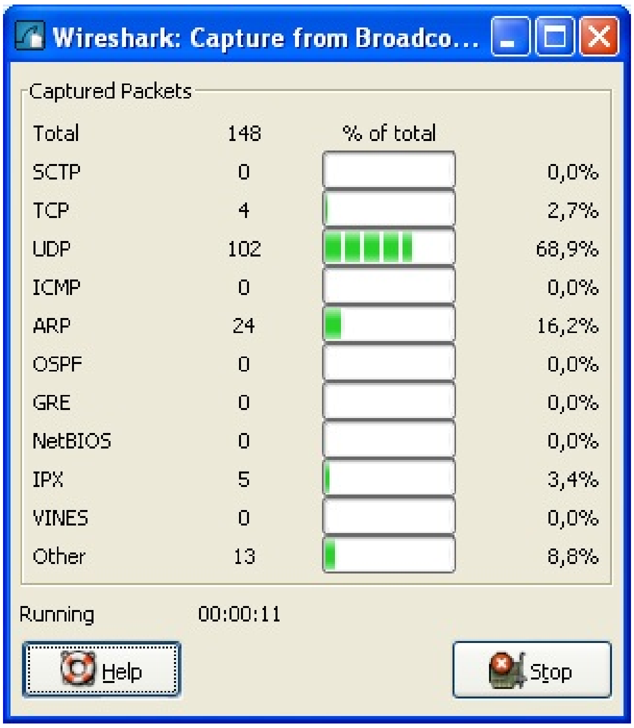
SESSIONNAME (terminal server)

```
&lt;remote name&gt;
```

4.9. 在捕捉过程中

捕捉时，会出现下面的对话框

图 **4.3.** 捕捉信息对话框



上述对话框会向你显示捕捉到包的数目，捕捉持续时间。选择的被统计的协议无法更改(什么乌意思？)

> 提示
>
> 这个对话框可以被隐藏，在前次的捕捉选项对话框设置"Hide capture info dialog box"即可。

4.9.1. 停止捕捉

运行中的捕捉线程可以用下列方法停止：

1. 使用捕捉信息对话框上的"stop"按钮停止。

   | | 注意 | |

   捕捉信息对话框有可能被隐藏，如果你选择了"Hide capture info dialog"

   |

2. 使用菜单项"Capture/ Stop"
3. 使用工具栏项" Stop"
4. 使用快捷键:Ctrl+E
5. 如果设置了触发停止的条件，捕捉达到条件时会自动停止。

4.9.2. 重新启动捕捉

运行中的捕捉进程可以被重新启动。这将会移出上次捕捉的所有包。如果你捕捉到一些你不感兴趣的包，你不想保留它，这个功能十分有用。

重新启动是一项方便的功能，类似于停止捕捉后，在很短的时间内立即开始捕捉。以下两种方式可以实现重新启动捕捉:

1. 使用菜单项"Capture/ Restart"
2. 使用工具栏项" Restart"

第 5 章 文件输入／输出及打印

目录

5.1. 说明

本章将介绍捕捉数据的输入输出。

- 打开／导入多种格式的捕捉文件
- 保存／导出多种格式的捕捉文件
- 合并捕捉文件
- 打印包

5.2. 打开捕捉文件

Wireshark可以读取以前保存的文件。想读取这些文件，只需选择菜单或工具栏的："File/ Open"。Wireshark将会 弹出打开文件对话框。详见第 5.2.1 节 “打开捕捉文件对话框”

如果使用拖放功能会更方便

要打开文件，只需要从文件管理器拖动你想要打开的文件到你的Wireshark主窗口。但拖放功能不是在所有平台都支持。

在你载入新文件时，如果你没有保存当前文件，Wireshark会提示你是否保存，以避免数据丢失。(你可以在首选项禁止提示保存)

除Wireshark原生的格式(libpcap 格式，同样被 tcpdump/Windump和 其他基于libpcap/WinPcap使用)外，Wireshark可以很好地读取许多捕捉文件格式。支持的格式列表见第 5.2.2 节 “输入文件格式”

5.2.1. 打开捕捉文件对话框

打开文件对话框可以用来查找先前保存的文件。表 5.1 “特定环境下的打开文件对话框”显示了一些Wireshark打开文件对话框的例子。

对话框的显示方式取决于你的操作系统

对话框的显示方式取决于操作系统，以及GTK+工具集的版本。但不管怎么说，基本功能都是一样的。

常见对话框行为：

- 选择文件和目录
- 点击Open/OK按钮，选择你需要的文件并打开它
- 点击Cancle按钮返回Wireshark主窗口而不载入任何文件。

Wireshark对话框标准操作扩展

- 如果选中文件，可以查看文件预览信息(例如文件大小，包个数。。。)
- 通过"filter:"按钮、显示字段指定显示过滤器。过滤器将会在打开文件后应用。在输入过滤字符时会进行语法检查。如果输入正确背景色为绿色，如果错误或输入未结束，背景色为绿色。点击filter按钮会打开过滤对话框，用于辅助输入显示过滤表达式。(详见第 6.3 节 “浏览时过滤包”)

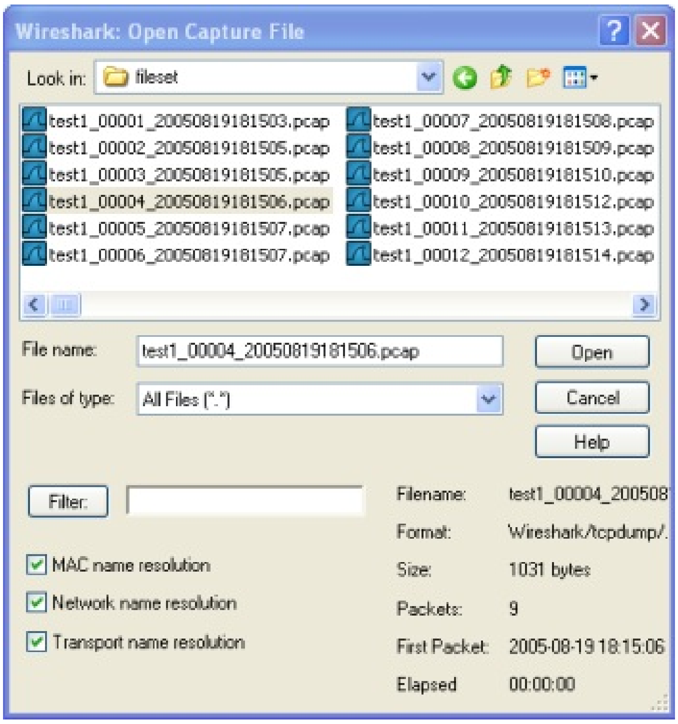
XXXX-we need a better description of these read filters(貌似说这一段需要更多的做介绍)

- 通过点击复选框指定那些地址解析会被执行。详见第 7.6 节 “名称解析”

> 在大文件中节约大量时间 |
>
> 你可以在打开文件后修改显示过滤器，和名称解析设置。但在一些巨大的文件中进行这些操作将会占用大量的时间。在这种情况下建议在打开文件之前就进行相关过滤，解析设置。

表 5.1. 特定环境下的打开文件对话框

图 **5.1. Windows**下的打开对话框



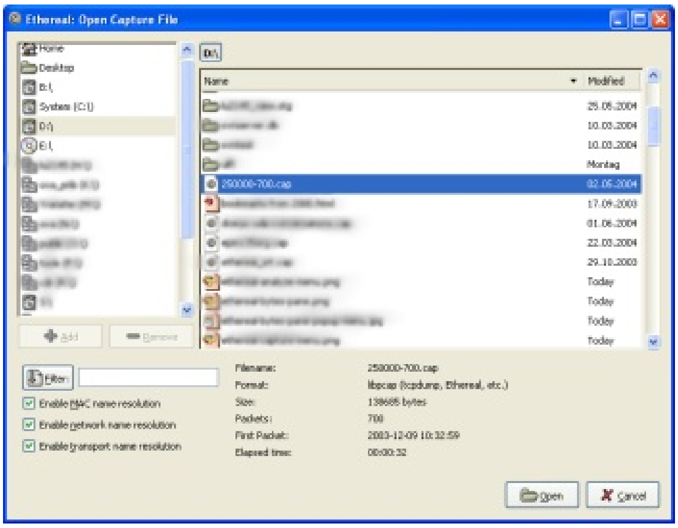
Microsoft Windows(GTK2 installed)

此对话框一般都带有一些wireshark扩展

此对话框的说明：

- 如果可用，"help"按钮将会打开本节的用户手册。
- "Filter."按钮 在当前版本的windows下不可用(我看了一下，的确不可用，但过滤输入框还是可用的)
- 错误提示功能:如果Wireshark无法识别选中的捕捉文件，Open按钮将为灰色不可用[a]

图 **5.2.** 新版**GtK**下的打开对话框



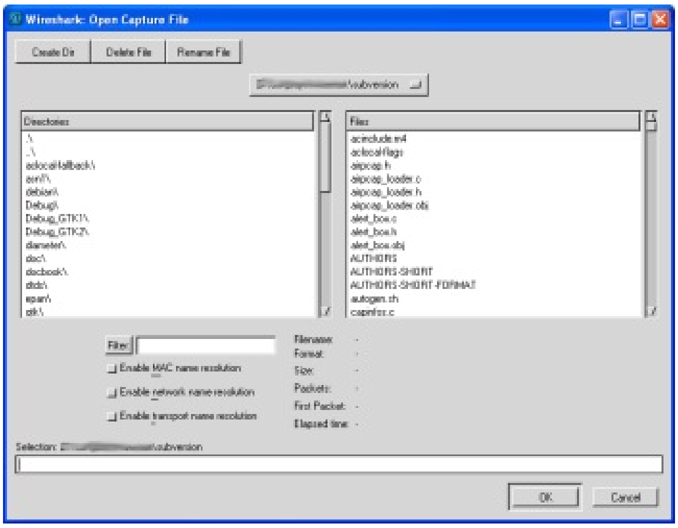
Unix/Linux:GTK version >= 2.4

这是在Gimp/GNOME桌面环境下的打开文件对话框

对此对话框的说明。

- “+”按钮可以将右侧选中的目录添加到收藏夹。成为预设目录。
- "-"按钮可以移除左侧目录列表中选中的目录。("Home","Desktop","Filesystem"不可以移除)
- 如果Wireshark不能识别选中的捕捉文件，"Open"按钮将是灰色不可用。

图 **5.3.** 旧版**GTK**下的打开对话框



Unix/Linux: GTK version < 2.4 / Microsoft Windows (GTK1 installed)

gimp/gnome桌面环境，或windows gtk1下的的。

该对话框说明

- 如果未能识别不做文件，Open按钮将为灰色不可用

[a] 我测试了一下，无论什么文件，Wireshark都会去尝试打开，更遑论错误检查

5.2.2. 输入文件格式

可以打开的捕捉文件格式列表：

- libpcap, tcpdump and various other tools using tcpdump's capture format
- Sun snoop and atmsnoop
- Shomiti/Finisar Surveyor captures
- Novell LANalyzer captures
- Microsoft Network Monitor captures
- AIX's iptrace captures
- Cinco Networks NetXray captures
- Network Associates Windows-based Sniffer and Sniffer Pro captures
- Network General/Network Associates DOS-based Sniffer (compressed or uncompressed) captures
- AG Group/WildPackets EtherPeek/TokenPeek/AiroPeek/EtherHelp/PacketGrabber captures
- RADCOM's WAN/LAN Analyzer captures
- Network Instruments Observer version 9 captures
- Lucent/Ascend router debug output
- HP-UX's nettl
- Toshiba's ISDN routers dump output
- ISDN4BSD i4btrace utility
- traces from the EyeSDN USB S0
- IPLog format from the Cisco Secure Intrusion Detection System
- pppd logs (pppdump format)
- the output from VMS's TCPIPtrace/TCPtrace/UCX$TRACE utilities
- the text output from the DBS Etherwatch VMS utility
- Visual Networks' Visual UpTime traffic capture

- the output from CoSine L2 debug
- the output from Accellent's 5Views LAN agents
- Endace Measurement Systems' ERF format captures
- Linux Bluez Bluetooth stack hcidump -w traces
- Catapult DCT2000 .out files

> 不正确的包类型可能无法会导致打开错误。
>
> 某些类型的捕捉包可能无法读取。以太网环境下捕捉的大部分类型格式一般都等打开。但有些包类型(如令牌环环包)，不是所有的格式都被wireshark支持。

5.3. 保存捕捉包

你可以通过File->Save As...菜单保存捕捉文件。在保存时可以选择保存哪些包，以什么格式保存。

> 保存可能会丢失某些有用的信息
>
> 保存可能会少量都是某些信息。例如：已经被丢弃的包会丢失。详见???

5.3.1. "save Capture File As/保存文件为"对话框

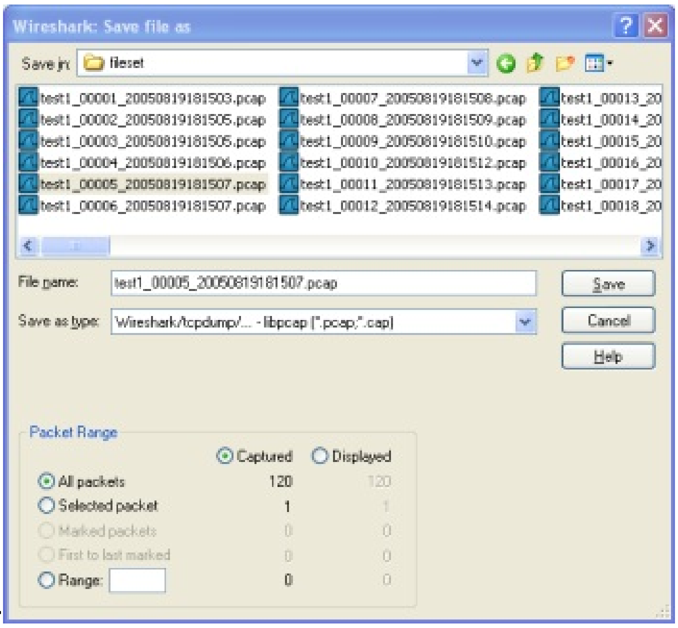
"Save Capture File As"对话框用于保存当前捕捉数据到文件。???列举了该对话框的一些例子。

> 对话框的显示方式取决于你的操作系统
>
> 对话框的显示方式取决于你的操作系统和GTK+工具集版本的不同。但大部分基本功能都是一样的。

表 **5.2. 特定环境下的"Save Capture File As"**对话框

|

图 **5.4. Windows**下的保存为对话框



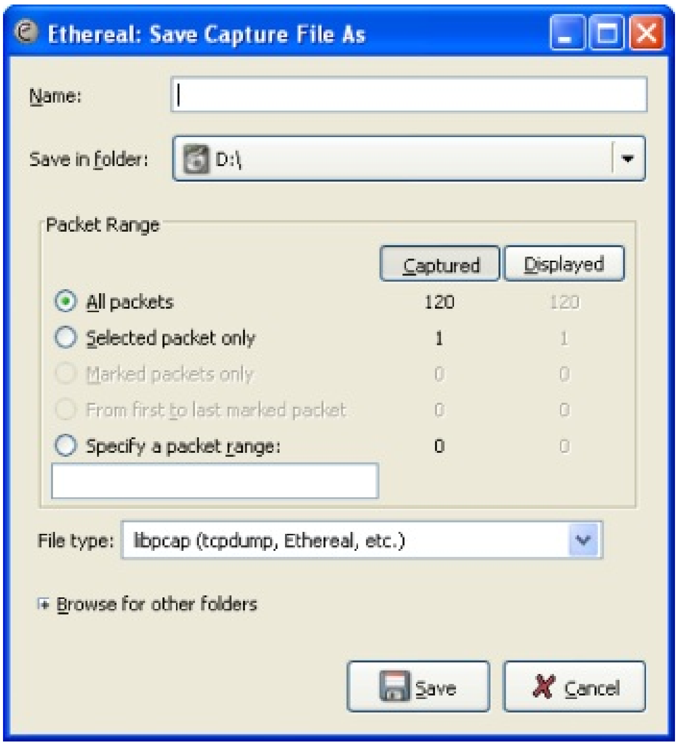
| **Microsoft Windows(GTK2 installed)**

此对话框一般都带有一些wireshark扩展

此对话框的说明：

- 如果可用，"help"按钮将会打开本节的用户手册。
- 如果你未输入文件扩展名-例如.pcap,Wireshark会自动添加该文件格式的标准扩展名。

图 **5.5.** 新版**GtK**下的保存为对话框



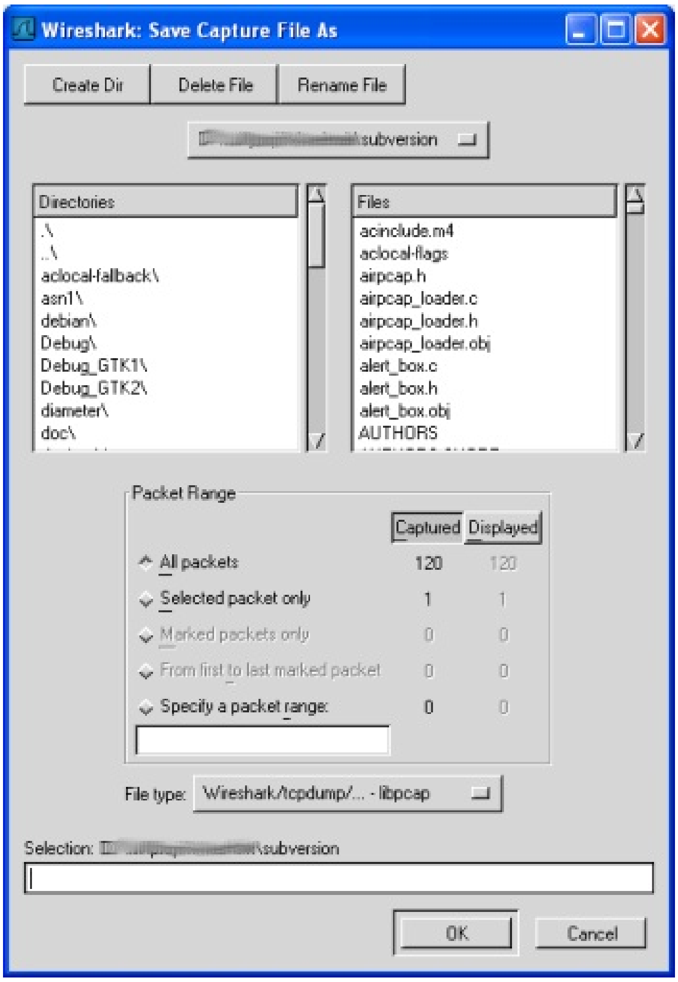
Unix/Linux:GTK version >= 2.4

这是在Gimp/GNOME桌面环境下的保存文件对话框

对此对话框的说明。

- "Browse for other flders"前的“+”按钮可以让你指定文件保存的位置。。

图 **5.6.** 旧版**GTK**下的保存为对话框



Unix/Linux: GTK version < 2.4 / Microsoft Windows (GTK1 installed)

gimp/gnome桌面环境，或windows gtk1下的的。

通过这些对话框，你可以执行如下操作：

1. 输入你指定的文件名。
2. 选择保存的目录
3. 选择保存包的范围，见第 5.8 节 “包范围选项”
4. 通过点击"File type/文件类型"下拉列表指定保存文件的格式。见???

> 可供选中的文件格式可能会没有那么多 |
>
> 有些类型的捕捉格式可能不可用，这取决于捕捉包的类型。
>
> 可以直接保存为另一种格式。]
>
> 你可以以一种格式读取捕捉文件，保存时使用另外一种格式(这句可能翻译有误。)

1. 点击"Save/OK"按钮保存。如果保存时遇到问题，会出现错误提示。确认那个错误提示以后，你可以重试。

2. 点击"Cancel"按钮退出而不保存捕捉包。

5.3.2. 输出格式

可以将Wireshark不着的包保存为其原生格式文件(libpcap)，也可以保存为其他格式供其他工具进行读取分析。

> 各文件类型之间的时间戳精度不尽相同
>
> 将当前文件保存为其他格式可能会降低时间戳的精度，见第 7.3 节 “时间戳”

Wireshark可以保存为如下格式。

- libpcap, tcpdump and various other tools using tcpdump's capture format (.*pcap*,.cap,*.dmp)
- Accellent 5Views (*.5vw)
- HP-UX's nettl (.*TRC0*,.TRC1)
- Microsoft Network Monitor - NetMon (*.cap)
- Network Associates Sniffer - DOS (.*cap*,.enc,.*trc*,fdc,*.syc)
- Network Associates Sniffer - Windows (*.cap)
- Network Instruments Observer version 9 (*.bfr)
- Novell LANalyzer (*.tr1)
- Sun snoop (.*snoop*,.cap)
- Visual Networks Visual UpTime traffic (.)

5.4. 合并捕捉文件

有时候你需要将多个捕捉文件合并到一起。例如：如果你对多个接口同时进行捕捉，合并就非常有用(Wireshark实际上不能在同一个实体运行多次捕捉，需要开启多个Wireshark实体)

有三种方法可以合并捕捉文件：

- 从"File"菜单使用，**menu item "Merge"(菜单项 "合并")**，打开合并对话框，见第 5.4.1 节 “合并文件对话框”
- 使用拖放功能，将多个文件拖放到主窗口。Wireshark会创建一个临时文件尝试对拖放的文件按时间顺序进行合并。如果你只拖放一个文件，Wireshark可能(只是)简单地替换已经打开的文件。
- 使用**mergecap**工具。该工具是在命令行进行文件合并的。它提供了合并文件的丰富的选项设置。见???

5.4.1. 合并文件对话框

通过该对话框可以选择需要合并的文件，并载入合并它们。

> 首先你会被提示有一个文件未保存
>
> 如果当前文件未保存，Wireshark会在启动合并对话框之前提示你是否保存文件。

此处的对话框的大多数内容与"Open Capture Files/打开捕捉文件"对话框类似，参见第 5.2.1 节 “打开捕捉文件对话框”

合并对话框中用于合并的控制选项：

将包插入已存在文件前

将选择文件内的包插入到当前已经载入文件之前

按时间顺序合并文件

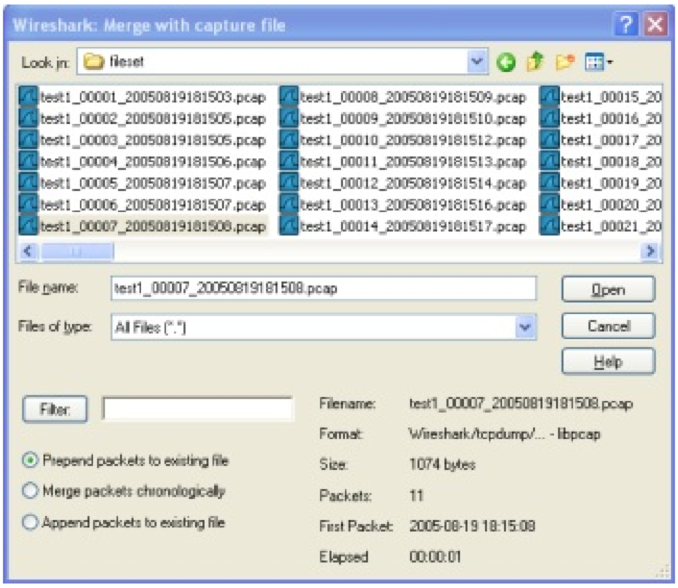
将当前选择的文件和已载入的文件里的所有包按时间顺序合并

追加包到当前文件

将选择文件的包插入到当前载入文件的末尾

表 5.3. 不同环境下的**"Merge Capture File As"**对话框

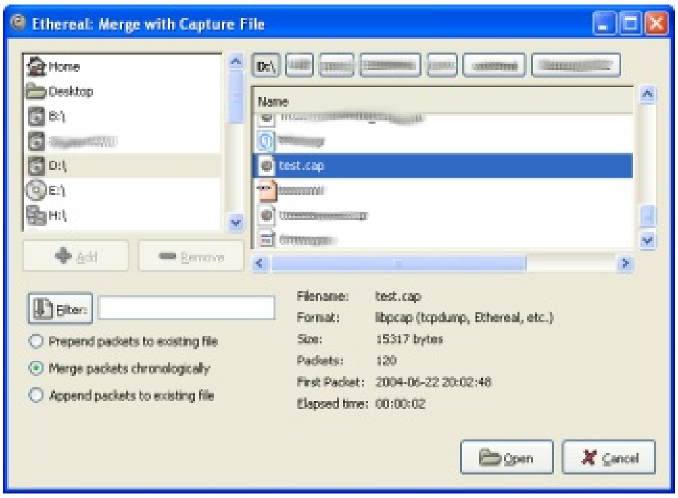
图 **5.7. Windows**下的**"合并"**对话框



Microsoft Windows(GTK2 installed)

此对话框一般都带有一些wireshark扩展

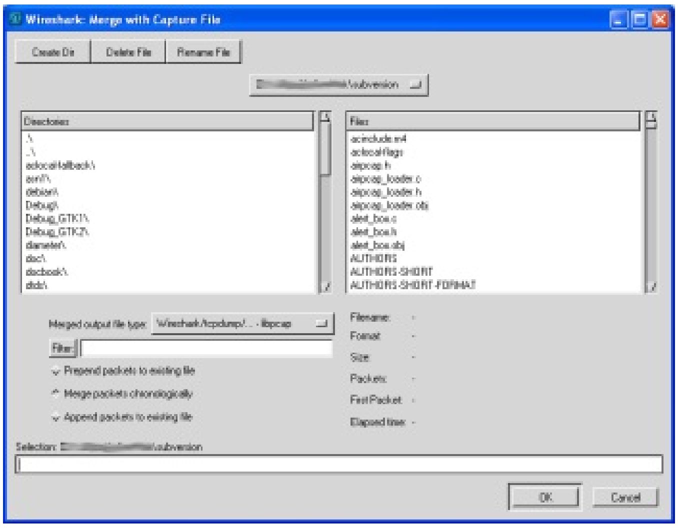
图 **5.8.** 新版**GtK**下的合并话框



Unix/Linux:GTK version >= 2.4

这是在Gimp/GNOME桌面环境下的合并对话框

图 **5.9.** 旧版**GTK**下的合并对话框



Unix/Linux: GTK version < 2.4 / Microsoft Windows (GTK1 installed)

5.5. 文件集合

在进行捕捉时(见：第 4.6 节 “捕捉文件格式、模式设置”)如果设置"Multiple Files/多文件"选项，捕捉数据会分割为多个文件，称为文件集合.

大量文件手动管理十分困难，Wirreshark的文件集合特性可以让文件管理变得方便一点。

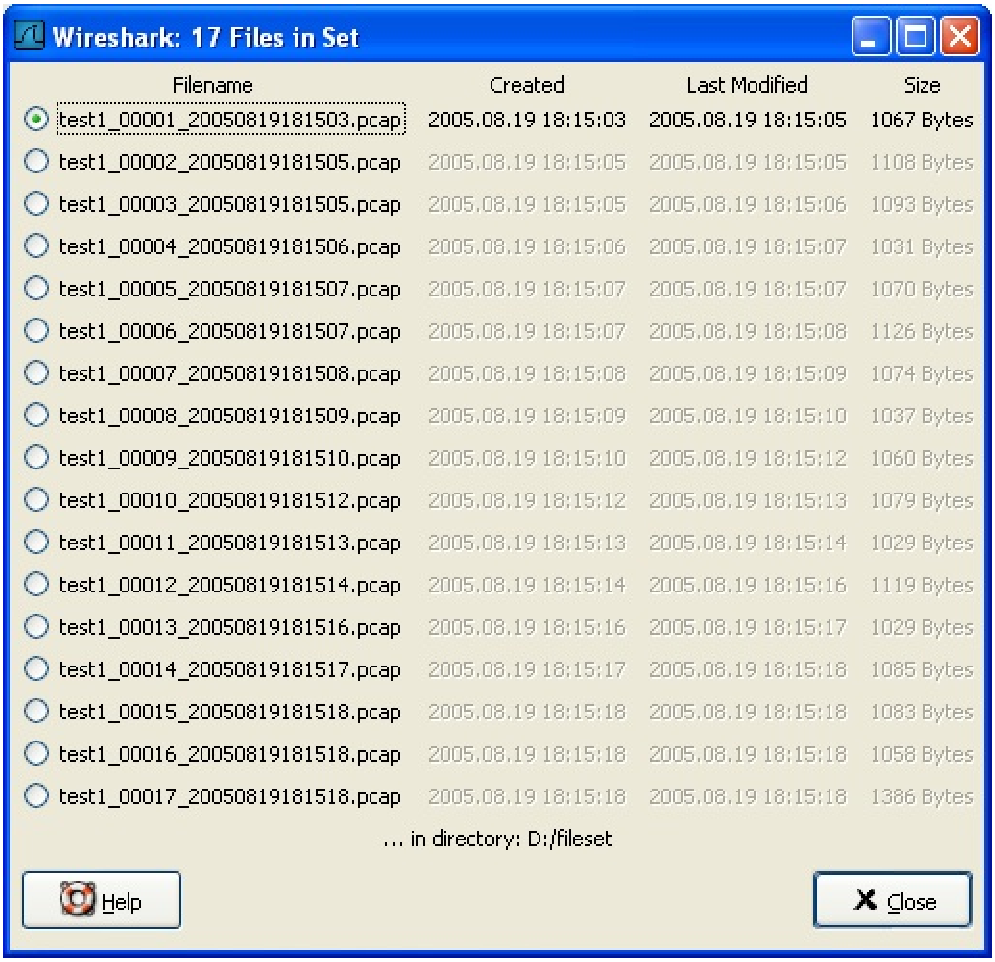
> **Wireshark是如何知道一个文件所属的文件集合t的?**
>
> 文件集合中的文件名以前缀号码+""+号码+""+日期时间+后缀的形式生成的。类似于："test_00001_20060420183910.pcap".文件集合所有的文件都有一个共同的前缀(例如前面的"test")和后缀(例如:".pcap")以及变化的中间部分。
>
> 要查找一个文件集合的所有文件。Wireshark会扫描当前载入文件的目录下的所有文件，找到那些和当前文件名具有相同部分（前缀和后缀）的作为文件集合。
>
> 这个简单的机制通常能正常运行，但也有它的弊端。如果几次进行的捕捉具有相同的前缀和后缀，Wireshark会将它们看作同一个文件集合。如果文件被更名或者放在不同的目录下，这样的按文件名查找机制会无法找到文件集合的所有文件。

使用"File"菜单项的子菜单"File Set"可以对文件集合集合进行很方便的控制。

- **List Files** 对话框显示一个对话框列出所有被识别出来属于当前文件集合的文件列表。
- **Next Files** 关闭当前文件，打开文件集合列表中的下一个文件。
- **Previous Files** 关闭当前文件，打开文件集合列表中的前一个文件。

5.5.1. 文件列表对话框

图 **5.10.** 文件列表对话框



每行包含文件集合中的一个文件的相关信息。

- **Filename** 文件名称。如果你双击文件名称(或者单击单选钮)，当前文件会被关闭，同时载入对应的文件。
- **Created** 文件创建时间。
- **Last Modified** 最后一次修改文件的时间。
- **size** 文件的大小。

最后一行"...indirctory:"显示所有文件所在的目录。

在每次捕捉文件被打开、关闭时，对话框的内容会变化。

Close按钮关闭该对话框。

5.6. 导出数据

Wireshark支持多种方法，多种格式导出包数据，本节描述Wireshark常见的导出包数据方法。

注意

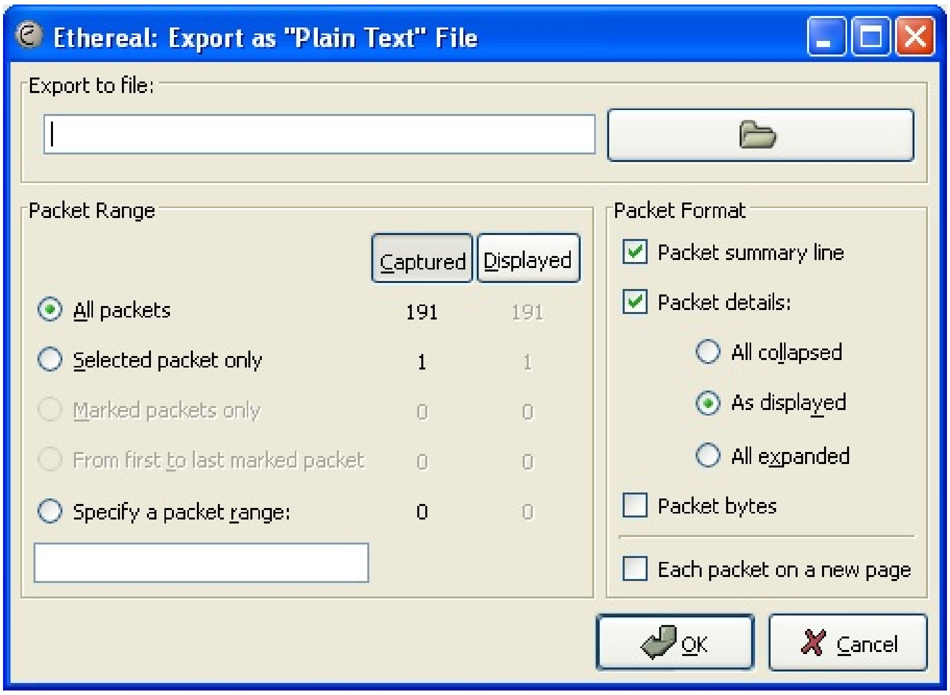
个别数据需要使用许多特殊方式导出，在合适的时候我们会对其进行介绍。

XXX - add detailed descriptions of the output formats and some sample output, too./同样需要对导出格式进行介绍，同样也需要一些范例范例

5.6.1. "Export as Plain Text File"对话框

导出包数据为"plain AscII "文本文本见，适合打印包数据。

图 **5.11. "Export as Plain Text File"**对话框



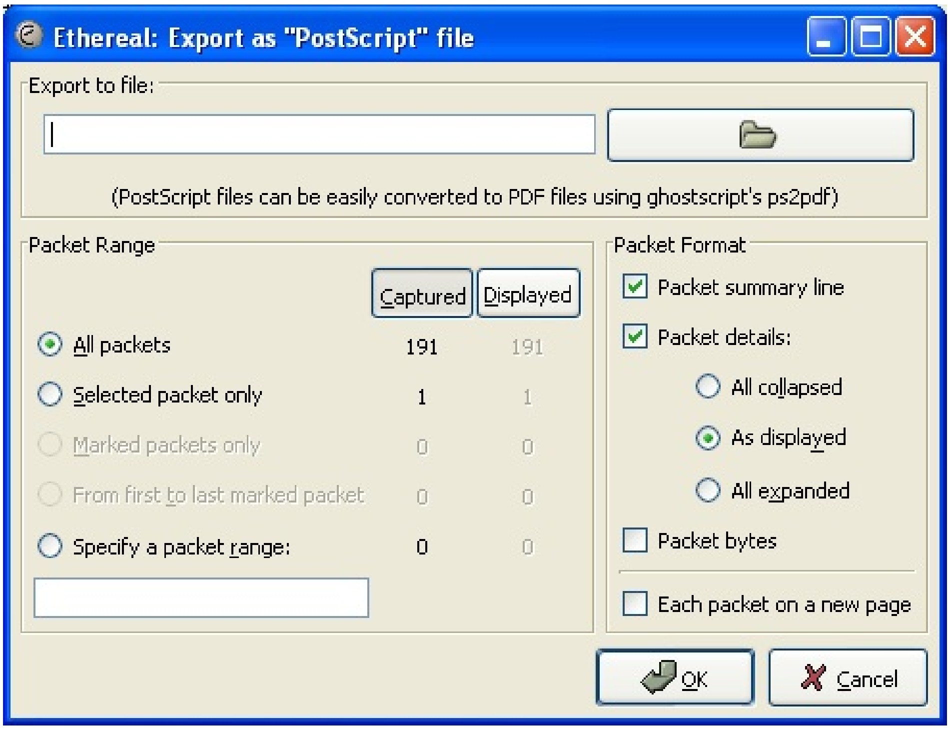
- **Export to file:**导出包数据为指定的文件
- **Packet Range** 参见第 5.8 节 "包范围选项"
- **Packet Details** 参见???

5.6.2. "Export as PostScript File" 对话框

导出数据为PostScript格式，PostScript是一种打印格式。

> 提示
>
> PostScribt文件可以使用ghostscrip转换为PDF格式。例如导出文件名为foo.ps，然后调用**ps2pdf foo.ps**命令就可以进行转换。

图 **5.12. "Export as PostScript File"** 对话框



- Export to file: 导出包数据为指定的文件
- Packet Range: 参见第 5.8 节 “包范围选项”
- Packet Details: 参见???

5.6.3. "Export as CSV (Comma Separated Values) File" 对话框

注：笔者认为此处应该增加截屏，因为我的*xp*下界面与前图风格迥异，这里就不提供了

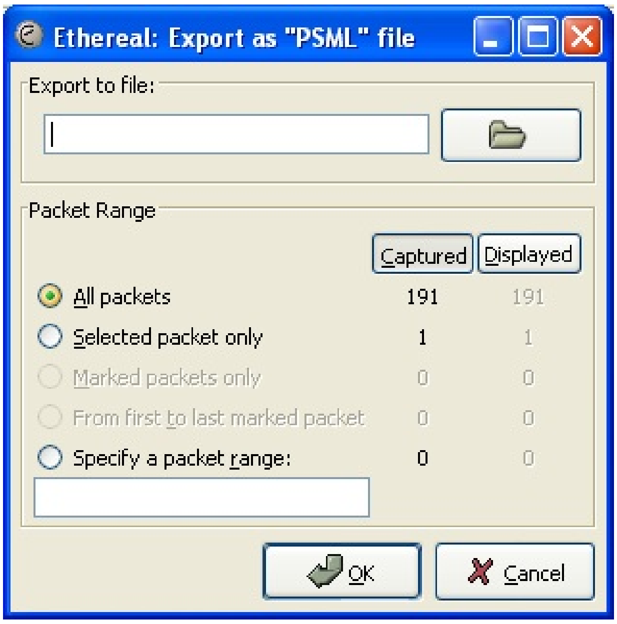
导出包的摘要为CVS格式，可以被电子表格程序使用。

- **Export to file** 导出包数据为指定的文件
- **Packet Range** 参见第 5.8 节 “包范围选项”

5.6.4. "Export as PSML File" 对话框

导出包数据为PSML格式，它是一种仅包含包摘要信息的xml格式。PSML格式的说明参见：http://www.nbee.org/Docs/NetPDL/PSML.htm.

图 **5.13. "Export as PSML File"**对话框



- Export to file:导出包数据为指定的文件
- Packet Range: 参见第 5.8 节 “包范围选项”

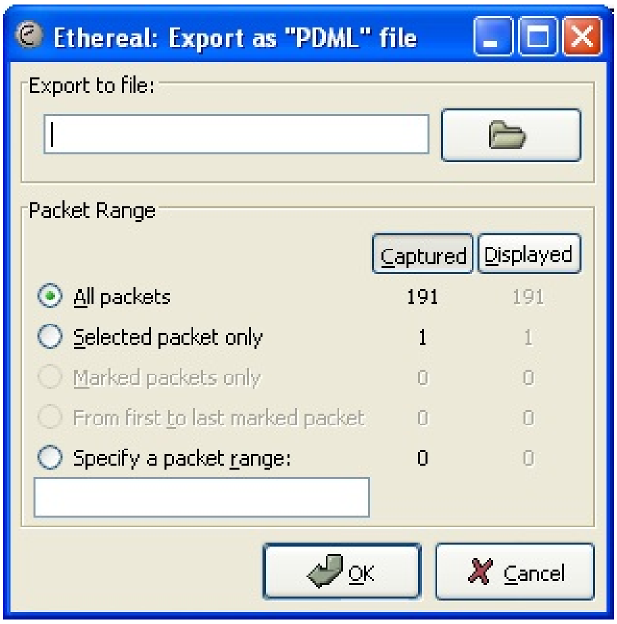
上图没有诸如Packet details的选项，因为PSML文件格式有特殊要求，不包含这些内容。

5.6.5. "Export as PDML File" 对话框

导出数据包为PDML格式，PDML是包含包详情的xml格式文件。PDML文件的说明见:http://www.nbee.org/Docs/NetPDL/PDML.htm

> 注意
>
> PDML格式还没有发行版，Wireshark执行PDML还处在测试阶段，期望外来版本会有所变化。

图 **5.14. "Export as PDML File"**对话框



- **Export to file**:将包数据导出到
- **Packet Range**: 参见第 5.8 节 “包范围选项”

上述对话框里没有诸如Packet details选项，这是由于PDML格式约定的内容决定的。

5.6.6. "Export selected packet bytes" 对话框

图 **5.15. "Export Selected Packet Bytes"** 对话框

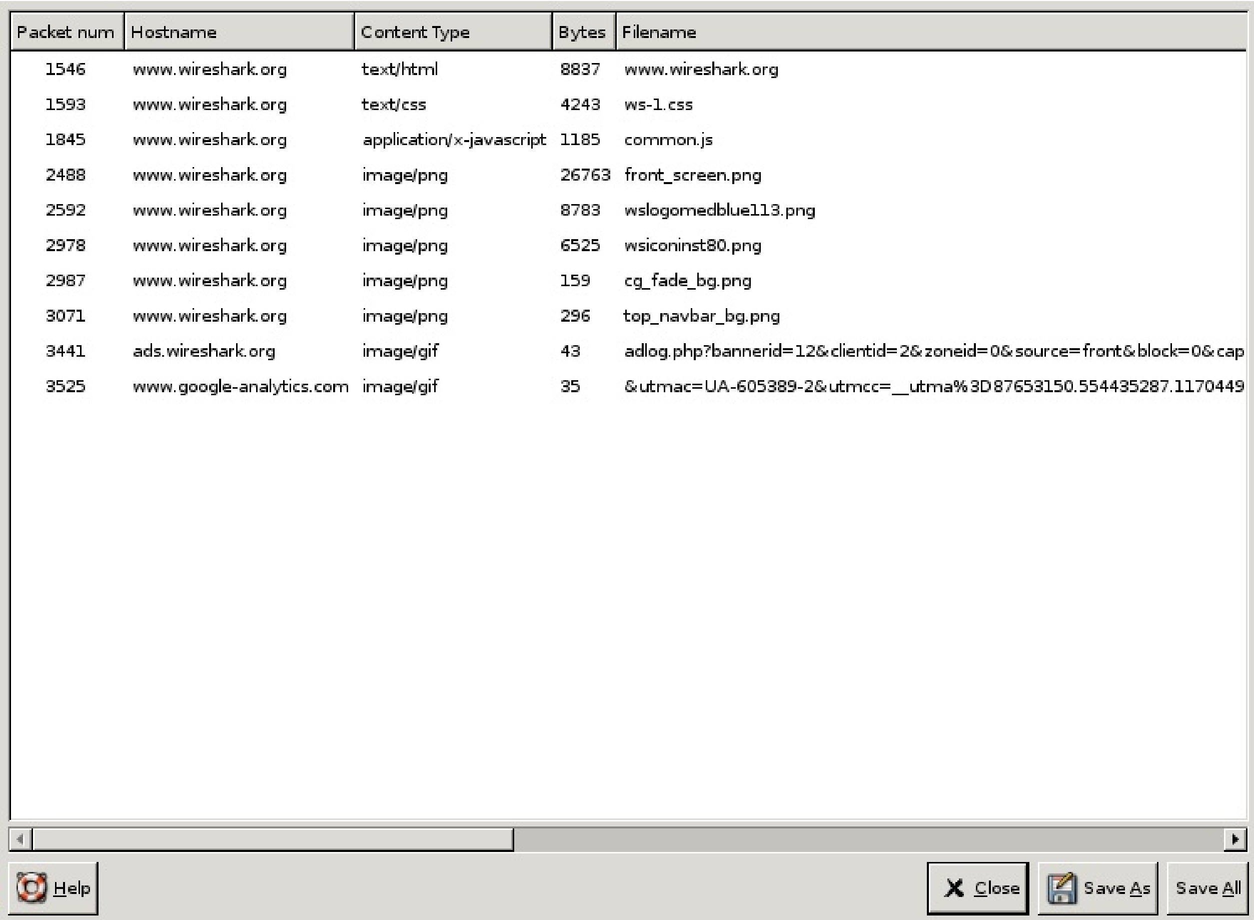


- **Name:** 导出数据包为文件
- **Save in folder:**导出数据包到指定目录
- **Browser for other folders** 通过浏览来指定导出数据的目录。

5.6.7. "Export Objects" 对话框

这个对话框是用来扫描当前打开包文件或者是正在捕捉中的包文件，将其中的对象，如HTML文档，图片文件，可执行文件等等任何可以通过HTTP传输的对象进行重组集合，让你可以将他们保存刀磁盘。如果捕捉正在进行中，列表会在发现新对象之后的几秒内立即更新。保存的对象不需要进行额外处理就可以被对应的查看工具打开，或者直接运行(如果它可以在Wireshark所在的平台运行的话)。这项功能在GTK1下的Wireshark中无法使用。

图 5.16. "Export Objects"对话框



各列说明

Packet num

包含该对象数据的包的数目，有时候多个对象可能包含在同一个包里。

Hostname

作为服务器相应HTTP请求发送对象的主机的主机名。

Content Type

对象的HTTP内容类型

Bytes

对象的字节数

Filename

URL的最后一部分(最后一个"/"之后)。通常这部分是文件名，但有时是一个又常又复杂的字符串，这通常表明该文件是一个"HTTP POST"请求。(类似于填写表单以后通过CGI提交后跳转页面的URL)

按钮说明：

Help

打开本节的用户手册(5.6.7节？)

Close

关闭该对话框

Save As

用指定文件名保存当前选择对象。默认文件名是filename列中显示的对象文件名。

Save All

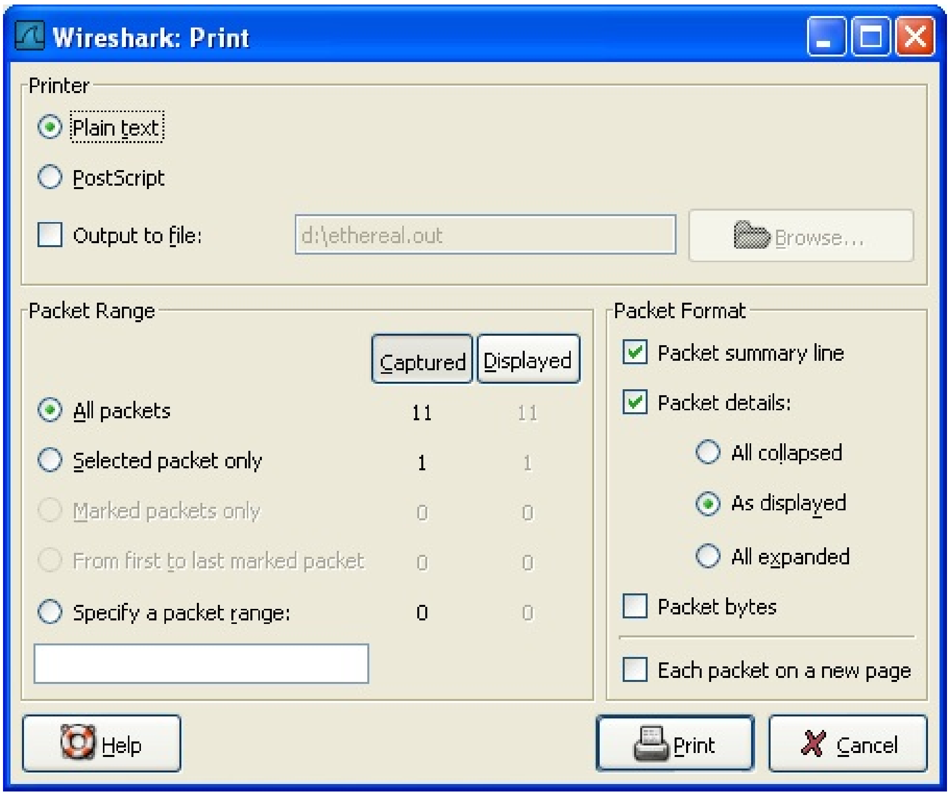
将列表中所有对象按filename列显示名称保存。系统会提示你选择哪个目录/文件夹保存他们。如果文件名在当前操作系统或者文件系统下不合法,Wireshark会提示错误，该对象不会被保存(但其他对象会被保存)。

5.7. 打印包

要打印包，选择File菜单的"Print..."菜单项。这时会弹出如图 5.17 “ "Print" 对话框”所示的打印对话框。

5.7.1. 打印 对话框

图 **5.17. "Print"** 对话框



下面的字段在打印对话可用。

Printer

该字段包括一对互斥的单选钮

Print Text

指定包打印为plain text格式

PostScript

在打印过程中使用PostScrtipt打印软件生成打印输出。[15]

Output to file

打印为文件，文件名使用输入的字段或者在浏览按钮选择。

如果你没有选择**Output to file**:复选框,你输入字段的地方或Browse。。按钮都是灰色。

Print command

设置打印时使用的命令

> 注意
>
> 打印命令在Windows平台不可用。

用于打印的命令通常是**lpr**.You would change it to specify a particular queue if you need to print to a queue other than the default.例如:

```
lpr -Pmypostscript
```

如果没有选中**Output to file**,该字段将是灰色不可用的。

Packet Range

需要要被打印的包，参见：第 5.8 节 “包范围选项”

Packet Format

选择输出文件的格式。你可以选择包以何种方式打印包。见图 5.19 “"Packet Format"选项卡”

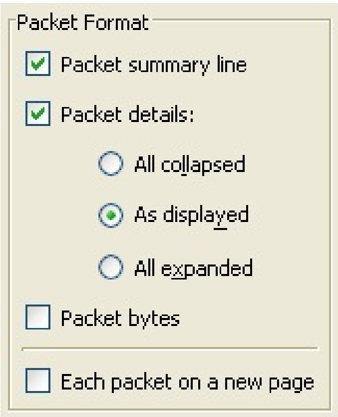
[15] 译者注：此处需要说明的是，如果没有打印机，或者不想打印，你应该在后面指定Output to file,指定打印输出未知， 另，out put to file输出的后缀名是.out,如果想用acrobat导入， 可以考虑将后缀名修改为.ps,这样可以被直接识别，当然，直接把文件拖放到Acrobat Distiller也可以直接生成PDF文件。

另：使用PostScript输出的文件具有良好的形式，比如在页首会加上列名，而直接打印为print text却没有这样的内容。

5.9. 包格式选项

包格式选项卡在很多输出对话框都能看到，它可以指定包的那些部分会被输出。

图 **5.19. "Packet Format"**选项卡



Packet summary line

导出包的摘要行，就是"Packet List"面板的内容

Packet Details

导出Packet Details树

All collaspsed

"Packet Details"面板在"all collapsed"状态下的所有信息(折叠所有分支)

As displayed

"Packet Details"面板当前状态下的信息

All expanded

"Packet Details"面板"all expanded"状态下的信息(展开所有分支)

Packet bytes

导出包字节，就是"Packet Bytes"面板的内容

Each Packet on a new page

输出是每个包单独一页(例如，如果保存/打印成text文件，会在包之间加上分节符)

第 6 章 处理已经捕捉的包

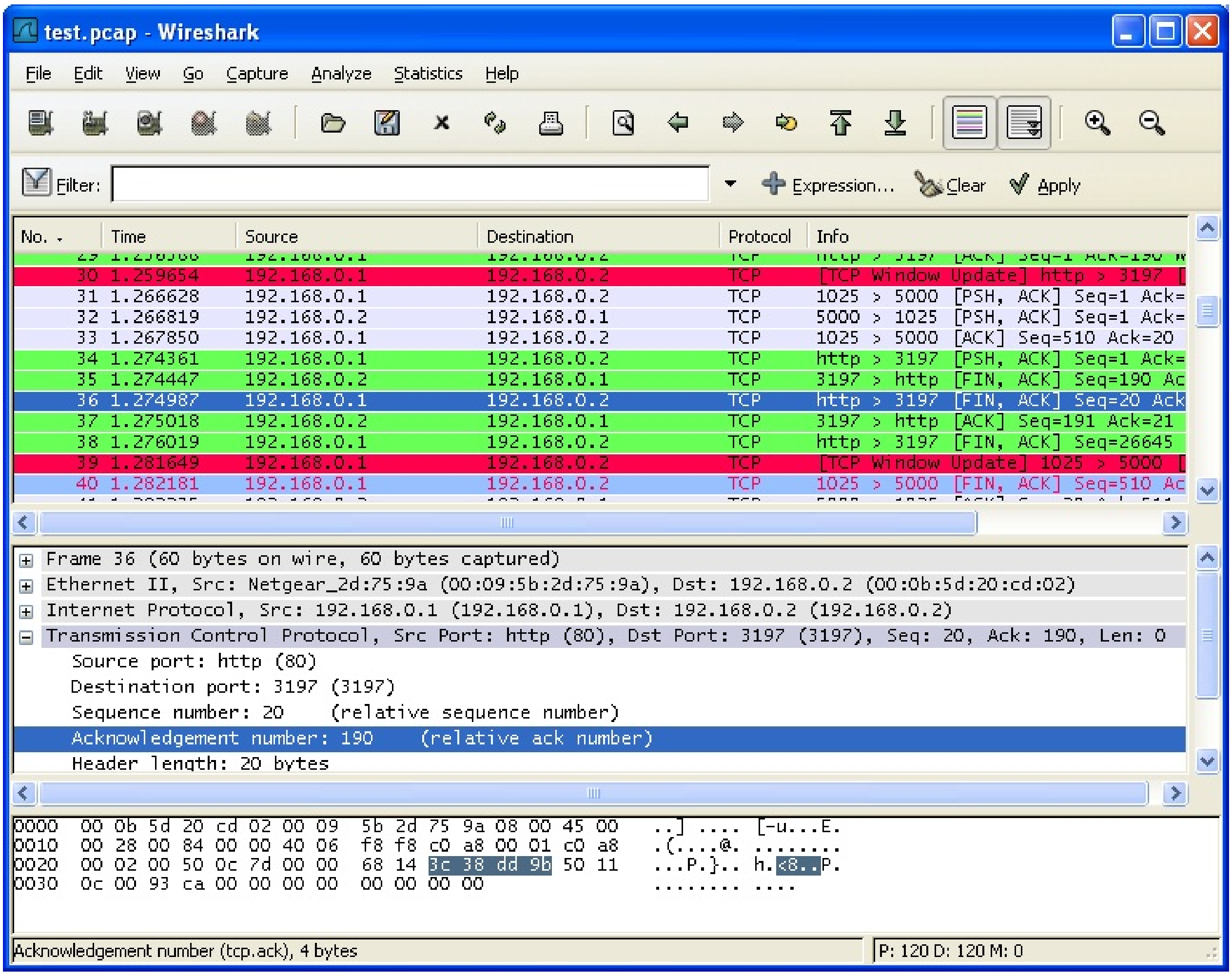
目录

6.1. 浏览您捕捉的包

在你已经捕捉完成之后，或者打开先前保存的包文件时，通过点击包列表面版中的包，可以在包详情面板看到关于这个包树状结构以及字节面版

通过点击左侧"+"标记,你可以展开树状视图的任意部分。你可以在面板点击任意字段来选择它。例如：在下图图 6.1 “Wireshark选择了一个TCP包后的界面”显示的就是选中TCP字段。同样可以选择ＴＣＰ包头的应答号(ack:190)，同时会出现在下方的字节浏览面版中。[16]

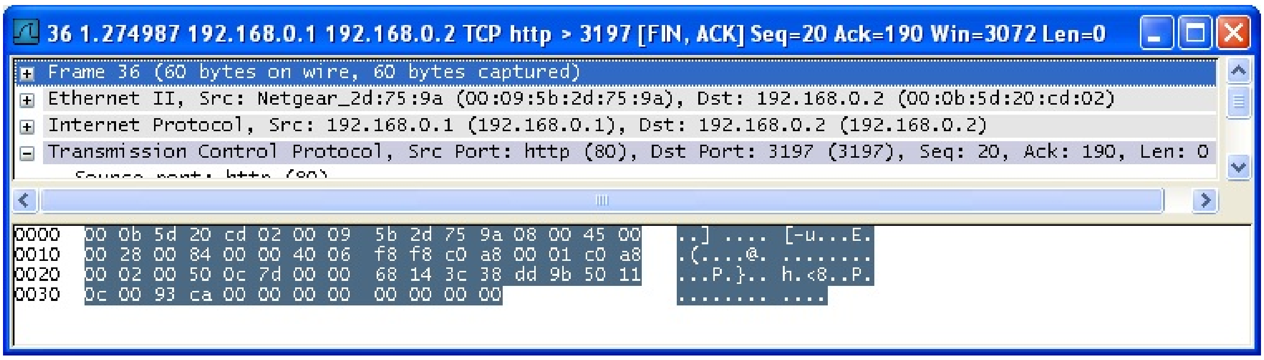
图 **6.1. Wireshark**选择了一个**TCP**包后的界面



在Wireshark正在捕捉时，您也可以进行同样的选择。(前提是您在捕捉选项对话框选择了实时更新列表(update list of packet in real time))

另外，您可以使用分离的窗口浏览单独的数据包，见图 6.2 “在分离窗口浏览包”，想要这样做，你只需要在选中包列表面版中您感兴趣的包，菜单 Display->Show Packet in New Windows 。它可以让你很轻松地比较两个或多个包。

图 **6.2.** 在分离窗口浏览包



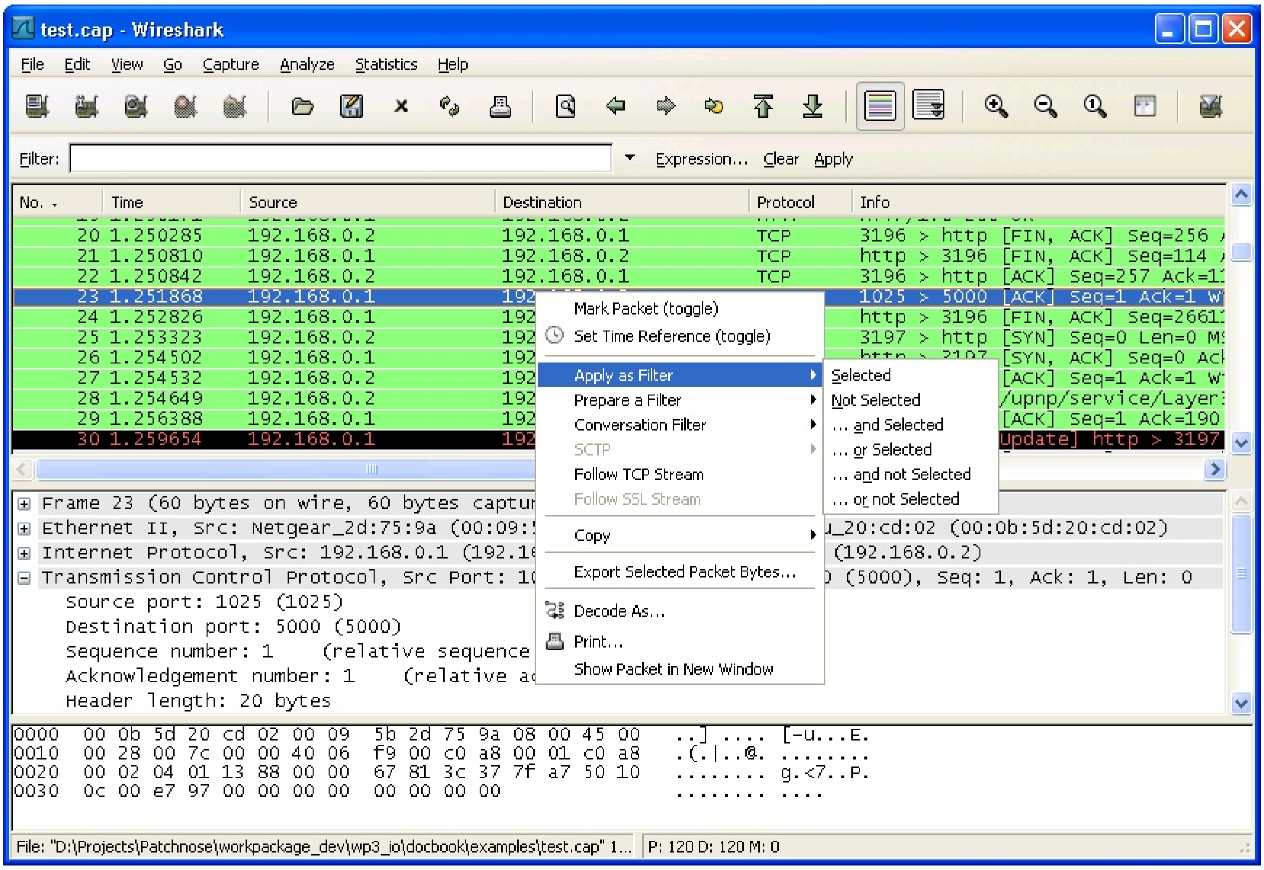
[16] 不甚了解下方的16进制转储怎么表达190的

6.2. 弹出菜单项

在包列表面板，包详情面板，包字节面板点击右键，都会出现对应的上下文弹出菜单

6.2.1. 包列表面板的弹出菜单

图 6.3. 包列表面板弹出菜单



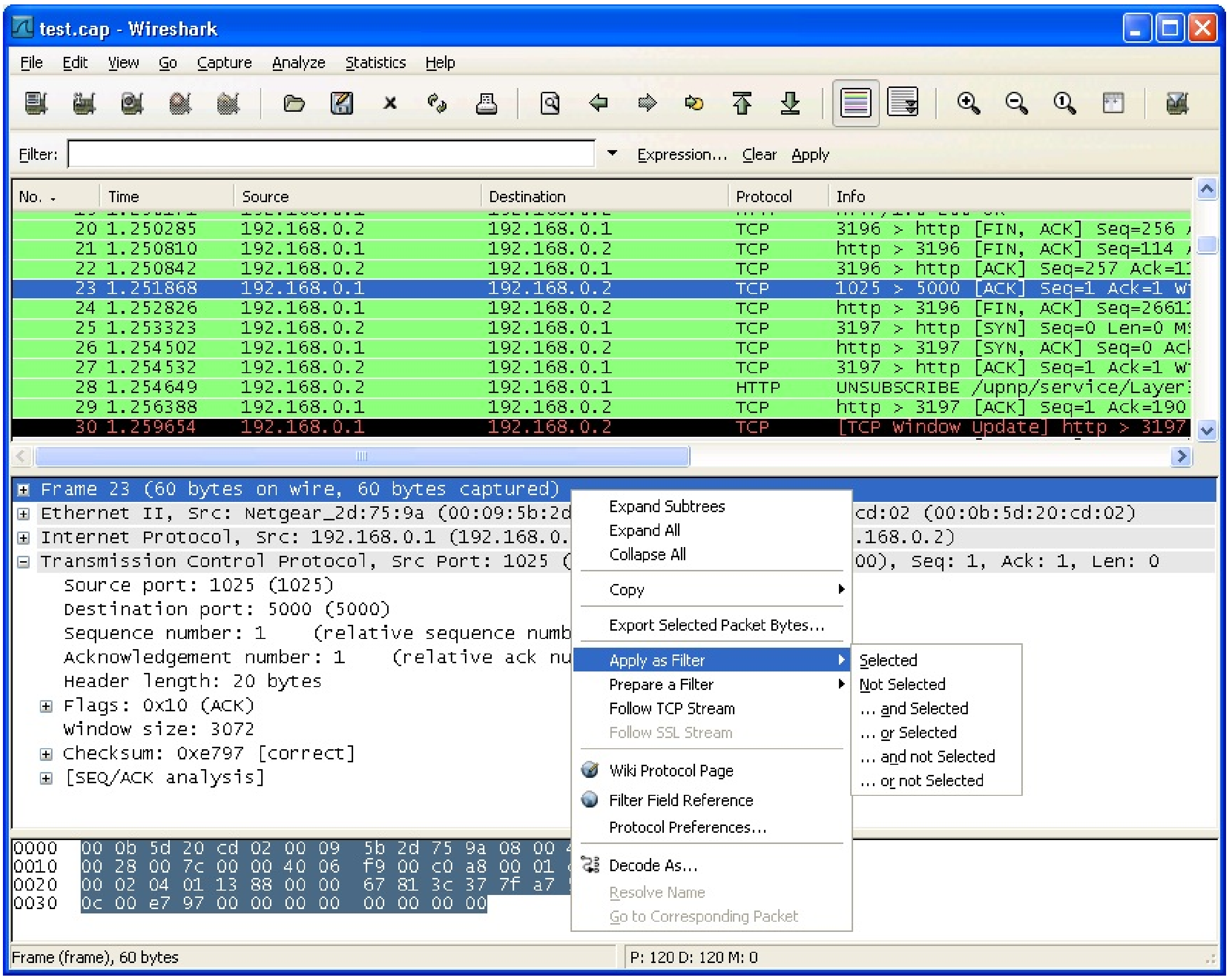
下表列出了该面版可用弹出菜单项的概述，主菜单能实现同样功能的菜单项，以及简短的描述。

表 6.1. 包列表弹出菜单项

项目	对应主菜单项	描述
Mark Packet(toggle)	Edit	标记/取消标记包
Set Time Reference(toggle)	Edit	设置/重设时间参考
Apply as Filter	Analyze	用当前选中的项作为过滤显示
Prepare a Filter	Analyze	准备将当前选择项作为过滤器
Conversation Filter	-	将当前选择项的地址信息作为过滤设置。选中该选项以后，会生成一个显示过滤，用于显示当前包两个地址之间的会话(不分源目标地址)。(XXX - add a new section describing this better.---作者似乎建议添加新章节详细描述)
STCP	-	有待补充
Follow TCP Stream	Analyze	浏览两个节点间的一个完整TCP流所有数据
Follow SLL Stream	Analyze	同上，将TCP替换成SSL理解
Copy/Summary(TEXT)	-	将摘要字段复制到剪贴板。(以tab分开的文本)
Copy/Summary(CVS)	-	将摘要字段复制到剪贴板，(CVS格式,逗号分开)
Copy/As Filter	-	以当前选择项，建立一个显示过滤器，复制到剪贴板
Copy/Bytes(Offset Hex Text)	-	以16进制转储格式将包字节复制到剪贴板。
Copy/Bytes(Offset Text)	-	以16进制转储格式将包字节复制到剪贴板。不包括文本部分。
Copy/ Bytes (Printable Text Only)	-	以ASCII码格式将包字节复制到剪贴板，包括非打印字符。
Copy/ Bytes (HEX Stream)	-	以16进制未分段列表数字方式将包字节复制到剪贴板，（an unpunctuated list of hex digits 应该有专有名词，有兴趣的查一下）
Copy/ Bytes (Binary Stream)	-	以raw binary格式将包字节复制到剪贴板。数据在剪贴板以"MIME-type application/octet-stream"存储，该功能在GTK+1.x环境下不支持
Export Selected Packet Bytes...	File	与文件菜单同名项目功能一样。允许将Raw packet 字节转换为二进制文件它
Decode As...	Analyze	在两个解析之间建立或修改新关联(不知所云)
Print...	File	打印包
Show Packet in New Window	View	在新窗口显示选中的包

6.2.2. 包详情面板的弹出菜单

图 **6.4.** 包详情面板弹出上下文菜单项



下表介绍了包详情列表菜单项的功能描述，及其他可以提供该功能的主菜单

表 **6.2.** 包详情面板弹出上下文菜档项

项目	对应的主菜单	描述
Expand Subtrees	View	展开当前选择的子树
Expand All	View	展开捕捉文件的所有包的所有子树
Collapse All	View	关闭包中所有已展开的子树
Copy/Description	-	复制选择字段显示的文本到剪贴板
Copy/AS Filter	Edit	将选择项目作为显示过滤内容复制到剪贴板
Copy/Bytes(Offset Hex Text)	-	将包字节以Hexdump-like格式存储到剪贴板；类似于包列表面板中同名的命令，但是拷贝结果仅仅是树分支中被选中部分(包字节面板中被选中字节)
Copy/Bytes(Offset		以Hexdump-linke格式保存到剪贴，不包括文本部

Copy/Bytes(Offset Hex)	-	分。类似于包列表命令，不同之处在于此处仅拷贝树分支选中部分(包字节面板选中部分)
COPY/Bytes (printable Text Only)	-	以ASCII格式拷贝包字节，非打印字符除外；类似于包列表面板中同样的命令。不同点在于此处仅拷贝选择的树分支（包字符被选择部分）
Copy/Bytes(Hex Stream)	-	j以unpunctuated list hex digits形式保存到剪贴板，类似于包列表面板中的命令，不同之处在于仅复制选中子树部分(包字节面板选中部分)
Copy/Bytes(Binary Stream)	-	以raw binary格式拷贝到剪贴板；类似于包列表面板中的命令，不同之处在于仅拷贝选中部分子树(包字节面板选中部分)。数据以MIME-type“Application/octet-stream”存储在剪贴板.该功能在GTK+1.x下不可用
Export Selected Packet Bytes...	File	同文件菜单中的同名项一样。导出raw packet 字节为二进制文件。
Apply as Filter	analyze	将当前选择项作为过滤内容，并应用
Preapare a Filter	Analyze	将当前选择项作为过滤内容，但不立即应用
Follow TCP Stream	Analyze	追踪两个节点见，被选择包所属TCP流的完整数据
Follow SSL Stream	Analyze	同上
Wiki Protocol Page	-	显示当前选择协议的对应WIKI网站协议参考页
Filter Field Reference	-	显示当前过滤器的WEB参考
Protocol Preferences...	-	如果协议字段被选中，点击该选项，打开属性对话框，选择对应协议的页面，???
Decode As...	Analyze	更改或应用两个解析器之间的关联(什么鸟意思？)
Resolve Name...	View	对选择的包进行名称解析，不是指所有的包
Go to corresponding Packet ...	Go	跳到当前选择包的相应包。

TNND,表格让人崩溃

6.3. 浏览时过滤包

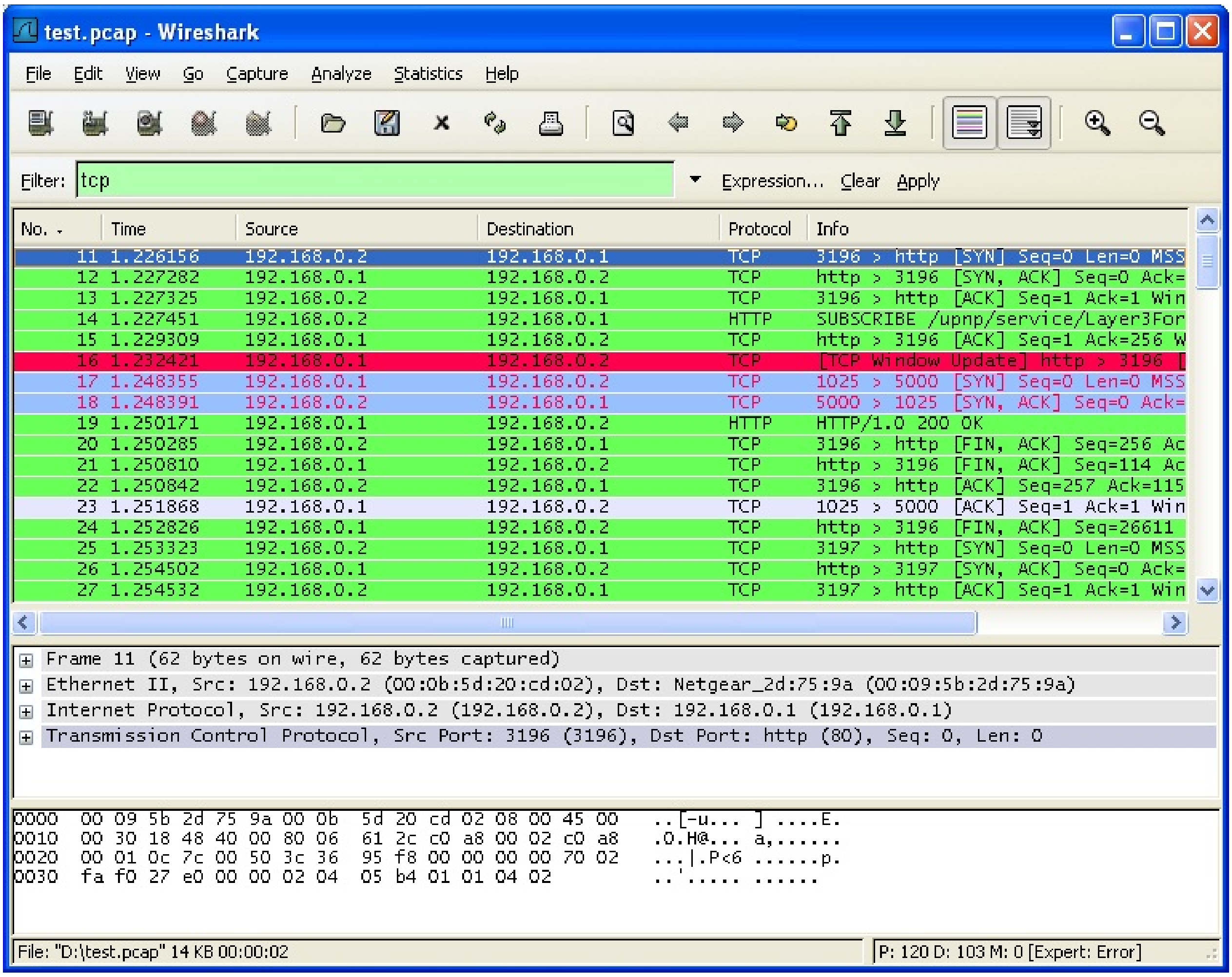
Wireshark有两种过滤语法：一种是捕捉包时使用，另一种是显示包时使用。本节介绍第二种过滤语法：显示过滤。第一种过滤语法在第 4.8 节 “捕捉时过滤”提到

显示过滤可以隐藏一些你不感兴趣的包，让你可以集中注意力在你感兴趣的那些包上面。你可以用从以下几个方面选择包:

- 协议
- 预设字段
- 字段值
- 字段值比较
- 。。。以及许多

根据协议类型选择数据报，只需要在**Filter**框里输入你刚兴趣的协议，然后回车开始过滤。???显示了你输入tcp进行过滤后的图。

图 **6.5.** 用**TCP**协议过滤



或许你没有注意到，上图显示的已经仅有TCP协议了（从图中可以看到1-10号包已经被隐藏）。因为包的编号是固定不变的，所以第一个包显示的编号是11。

注意

当你使用过滤时，所有的包依然保留在捕捉文件里。显示过滤只是更改捕捉文件的显示方式而非内容。

你只能对Wireshark可以识别的协议进行过滤。你也可以对解析器已经添加到树视图的字段进行过滤，但仅限于解析器已经为字段加上了协议缩写的。在**Add Expression...**对话框可以看到可用的字段列表.详见第 6.5 节 ““Filter Expression/过滤表达式”对话框”

例如：想要限制包列表面板仅显示来自或指向192.168.0.1的包，可以使用**ip.addr==192.168.0.**

注意

点击**Clear**可以移除过滤

6.4. 建立显示过滤表达式

Wireshark提供了简单而强大的过滤语法，你可以用它们建立复杂的过滤表达式。你可以比较包中的值，合并表达式为多个指定表达式。本节介绍了相关操作。

提示

你可以在Wireshark Wiki Display页找到发现大量的显示过滤范例。http://wiki.wireshark.org/DisplayFilters.

6.4.1. 显示过滤字段

包详情面板的每个字段都可以作为过滤使用。应用这些作为过滤将会仅显示包含该字段的包。例如：过滤字符串:TCP将会显示所有包含TCP协议的包。

通过"Help/Support Protocals"/帮助/协议支持菜单项访问"Display Filter Fields/显示过滤字段"可以查看完整完整的过滤字段列表。

6.4.2. 比较值

你可以通过在许多不同的比较操作建立比较过滤。详见表 6.3 “显示滤镜比较操作符”

提示

你可以使用下表中的英语和比较符(c-link)项达到同样的效果，它们也可以混合使用。

表 **6.3.** 显示滤镜比较操作符

English	C-linke	描述及范例
eq	==	**Equal** `ip.addr==10.0.0.5`
ne	!=	**Not equal** `ip.addr!=10.0.0.5`
gt	>	**Greate than** `frame.pkt_len>10`
lt	<	**Less than** `frame.pkt_len<128`
ge	>=	**Greater than or equal to** `frame.pkt_len >= 0x100`
le	<=	**Equal** `frame.pkt_len <= 0x20`

6.4.3. 组合表达式

你可以用逻辑操作符将过滤表达式组合在一起使用，见表 6.4 “显示过滤的逻辑操作符”

表 **6.4.** 显示过滤的逻辑操作符

<table>
<tr><th>English</th><th>C-linke</th><th>描述和范例</th></tr>
<tr><td>and</td><td>&&</td><td>Logical AND <code>ip.addr==10.0.0.5 and tcp.flags.fin</code></td></tr>
<tr><td>or</td><td>||</td><td>Logical OR <code>ip.addr==10.0.0.5 or ip.addr==192.1.1.1</code></td></tr>
<tr><td>xor</td><td>^^</td><td>Logical XOR <code>tr.dst[0:3] == 0.6.29 xor tr.src[0:3] == 0.6.29</code></td></tr>
<tr><td>not</td><td>!</td><td>Logical Not <code>not llc</code></td></tr>
<tr><td></td><td>[...]</td><td>Substring Operator Wireshark允许选择一个序列的子序列。在标签后你可以加上一对[]号，在里面包含用逗号(是不是冒号？)分离的列表范围。</td></tr>
<tr><td></td><td></td><td><code>eht.src[0:3] == 00:00:83</code></td></tr>
<tr><td></td><td></td><td>上例使用n:m格式指定一个范围。在这种情况下，n是起始位置偏移(0表示没有偏移，即是第一位，同理1表示向右偏移一位，便是第二位)，m是从指定起始位置的区域长度。</td></tr>
<tr><td></td><td></td><td><code>eth.src[1-2] == 00:83</code></td></tr>
<tr><td></td><td></td><td>上例使用n-m格式一个范围。在本例中n表示起始位置偏移,m表示终止位置偏移</td></tr>
<tr><td></td><td></td><td><code>eth.src[:4]=00:00:83:00</code></td></tr>
<tr><td></td><td></td><td>上例使用:m格式，表示从起始位置到偏移偏移位置m。等价于0:m</td></tr>
<tr><td></td><td></td><td><code>eth.src[4:]=20:20</code></td></tr>
<tr><td></td><td></td><td>上例使用n:格式，表示从最后位置偏移n个序列</td></tr>
<tr><td></td><td></td><td><code>eht.src[2] == 83</code></td></tr>
<tr><td></td><td></td><td>上例使用 n 形式指定一个单独的位置。在此例中中序列中的单元已经在偏移量n中指定。它等价于n:1</td></tr>
<tr><td></td><td></td><td><code>eth.src[0:3,102,:4,4:,2] == 00:00:83:00:83:00:00:83:00:20:20:83</code></td></tr>
<tr><td></td><td></td><td>Wireshark 允许你将多个分号隔开的列表组合在一起表示复合区域，如上例所示</td></tr>
</table>

6.4.4. 常见的错误

> 警告
>
> 在组合表达式中使用"!="操作符，像eth.addr,ip.addr,tcp.port,udp.port等元素可能会产生非预期效果

经常有人用**ip.addr ==1.2.3.4** 表达式来选择所有包含ip地址为1.2.3.4的包，

如果有人想用**ip.addr !=1.2.3.4** 表达式来排除ip地址为1.2.3.4的包，很不幸。它不会像你期待的那样。

相反，那个表达式为真值得条件是源地址或目标地址中的任意一个不等于1.2.3.4即可。因此，那个表达式**ip.addr !=1.2.3.4** 可以被读作："该包包含的ip字段值必须不为1.2.3.4"。因为一个ip数据报同含源地址和目标地址，只要两个地址有一个不为1,2,3,4表达式就为真。

接着上面的话题，如果你真想过滤捕捉文件中，ip地址包含1.2.3.4的包，正确的表达式应该是**!(ip.addr==1.2.3.4)**。它可以读作:"显示所有**'**字段名为**ip.addr**值存在**1.2.3.4'**为非真的包'",换句话说:"筛选所有字段名**ip.addr**的值中未出现**1.2.3.4**的包"

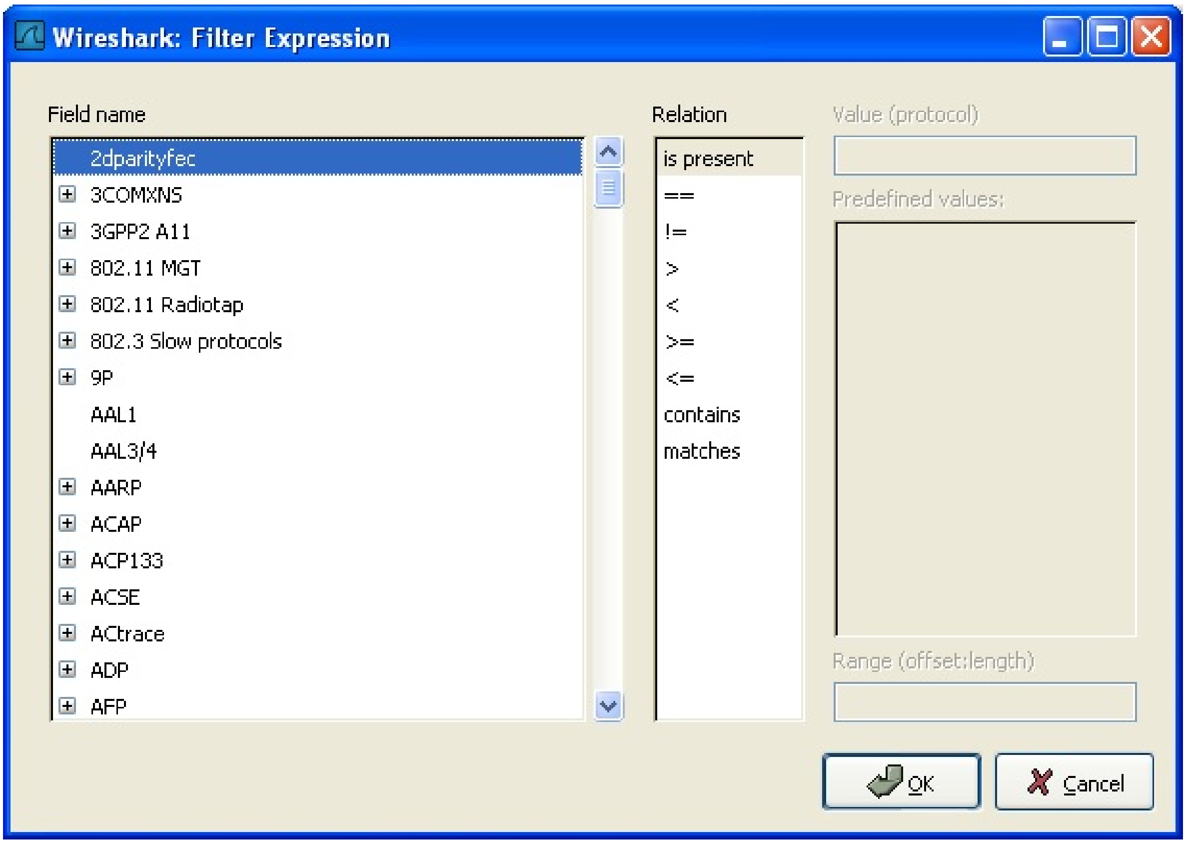
6.5. “Filter Expression/过滤表达式”对话框

当你熟悉Wireshark过滤系统，并了解你可以用那些标签进行过滤以后，你可以快速简单地输入过滤字符

但如果你是一个Wireshark新手，或者处理一些相对陌生的协议，你可能很难通过直接输入字符进行过滤。过滤表达式对话框会帮你解决这些问题

> 提示
>
> 过滤表达式对话框是学习输入表达式的不错的工具。(不知道用不错是不是有点委屈)

图 **6.6.** 过滤表达式对话框



打开上图的对话框以后。将会显示一个按协议类型分组的树分支列表，一个关系选择框。

Field Name

从协议字段树中选择协议字段。每个可过滤协议都放在第一级。点击+号展开列表，可以获得关于那些协议的可过滤字段。

Relation

从可用关系列表中选择关系。**is present**是一元关系，表示如果你选择的字段存在，表达式就为真值。其它关系都为二元关系，需要附加数据(例如：一个值来匹配)来完成。

如果你从字段名列表选择一个字段，并选择一个二元关系(例如等于关系"=="),你可能需要输入值，也有可能是范围信息。

Value

在此输入合适的配置值，输入的值同样要符合你选择的**field name**的属性值类型(例如 字符串).

Predefined values

有些协议字段包含预设值可用，这一点跟C语言中的枚举变量类似。如果选择的协议有这样的值定义，你可以在此选择。

Range

此处作者留空了

OK

如果你已经建立好了表达式，点击OK即可创建你的过滤字符串

Cancel

你可以点击**Cancle**按钮不做任何修改离开Add Expression。。。对话框。

6.6. 定义，保存过滤器

你可以定义过滤器，并给他们标记以便以后使用。这样可以省去回忆、重新输入某些你曾用过的复杂过滤器的时间。

定义新的过滤器或修改已经存在的过滤器有两种方法：1、在Capture 菜单选择**Capture Filters...**；2、在Analyze菜单选择**Display filter...**。Wireshark 将会弹出如图 6.7 “"捕捉过滤器"和"显示过滤器"对话框”所示话框。

> 注意
>
> 因为捕捉和显示滤镜的设定义和保存方式几乎完全一样。所以这里放在一起讲，二者之间的不同点会做标记
>
> 警告
>
> 你必须用**Save**来保存你的过滤器，**OK**或者**Apply**不会保存过滤器。关闭wireshark时会随之消失

图 6.7. "捕捉过滤器"和"显示过滤器"对话框



New

增加一个新的过滤器到列表中。当前输入的Filter name，Filter string值将会被使用。如果这些都为空，将会被设置为"new"(是说filtername还是说二者都是？)

Delete

删除选中的过滤器。如果没有过滤器被选中则为灰色

Filter name

修改当前选择的过滤器的名称

> 注意
>
> 过滤器名称仅用在此处为了区分方便而已，没有其他用处。你可以将多个过滤器使用同一个名称，但这样会很不方便

Filter string

修改当前选中过滤器的的内容。仅适用显示过滤：在输入时进行语法检查。

Add Expression

仅适用显示过滤：打开增加表达式对话框，辅助创建过滤表达式。详见第 6.5 节 ““Filter Expression/过滤表达式”对话框”

OK

仅适用显示过滤：为当前显示应用选择的过滤器，关闭当前对话框。

save

保存当前对话框设置。文件位置和格式见???

Close

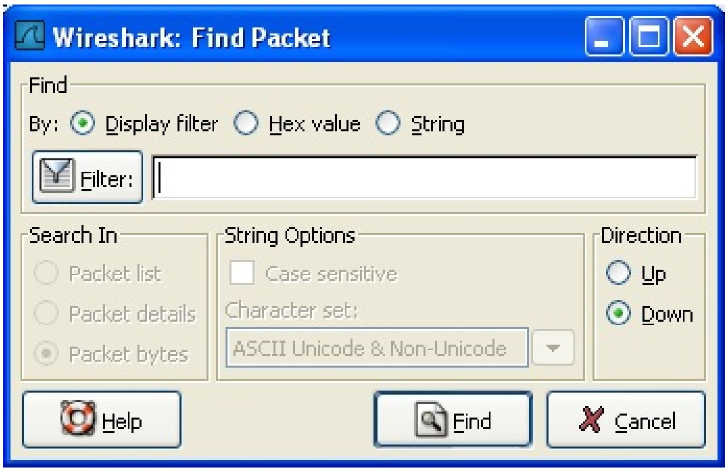
关闭当前对话框。将会放弃未保存的设置。

6.7. 查找包

当你捕捉到一些包以后，或者读取以前存储的包的时候，可以很容易的进行查找。从**Edit**菜单选择**Find Packet...**菜单项.Wireshark将会弹出图 6.8 “"Find Packet/查找包"对话框”所示对话框.

6.7.1. 查找包对话框

图 **6.8. "Find Packet/查找包"**对话框



首先你需要选择查找方式：

Display filter

在Filter:输入字段，选择查找方向，点击OK(过滤器方式)

例如：查找192.168.0.1发起的三步握手建立连接，使用如下字符:

```
ip.addr == 192.168.0.1 and tcp.flags.syn
```

显示过滤的详情，参见第 6.3 节 “浏览时过滤包”

Hex Value

在包数据中搜索指定的序列

例如，使用"00:00"查找下一个包含两个空字节的包数据。

String

在包中查找字符串，可以指定多种参数

输入的查找值将会被进行语法检查。如果语法检查无误，输入框背景色会变成绿色，反之则是红色。

你可以指定查找的方向通过：

UP

向上查找包列表（包编号递减方式）

Down

向下查找包列表(包编号递增方式)

6.7.2. "Find Next/查找下一个"命令

适用最后一次的查找设置继续查找

6.7.3. "Find Previous/查找上一个"命令

适用最后一次的设置修改查找方向，继续查找。

6.8. 到指定的包

通过"Go"菜单可以很轻松跳转到指定的包

6.8.1. "GO Back"返回命令

使用Go back返回包历史记录，工作方式跟web浏览器的页面历史记录类似

6.8.2. "Go Forward /向前"命令

前进到包的历史记录，工作方式跟web浏览器的页面历史记录类似

6.8.3. "Go to Packet/到指定的包"对话框

图 **6.9. "GO to packet/转到指定包"**对话框



输入包的编号，点击OK，跳转到指定的包(他妈的我怎么看都是jump to,怎么成了OK?).

6.8.4. "Go to Corresponding Packet/到对应的包"命令

如果被选择协议字段指向文件中的另一个包，该命令将会跳转到那个包。

> 注意
>
> 该协议字段看起来有点像超链接(就像浏览器里的)，双击该字段也可以跳转到对应的包。

6.8.5. "Go to Firest Packet/到第一个包"命令

跳到第一个包

6.8.6. "Go to Last Packet/到最后一个包"命令

跳到最后一个包

6.9. 标记包

你可以在包列表面板对包进行标记。被标记的包背景色为黑色，不管原来设置的颜色是怎样的。标记包有助于分析大的包文件时进行查找。

> 警告
>
> 包标记并没有存储在捕捉文件中或任何其他地方，关闭文件后，所有标记将会丢失。

在保存/导出/打印包时，你可以使用包标记控制输出包。标记包以后，可以输出进行区间选择。见第 5.8 节 “包范围选项”

对标记包可以进行三项操作

- **Mark packet(toggle)** 冻结以标记的单个包
- **Mark all packets** 标记所有包.

- **Unmark all packets** 取消所有标记

这些标记功能出现在"Edit"菜单。“Mark packet(toggle)”功能在弹出包列表面板弹出上下文菜单同样可以找到。

6.10. 时间显示格式及参考时间

在捕捉包的过程中，每个包都带有时间戳。时间戳会被保存在捕捉文件中，以备将来分析用。

关于时间戳，时区以及相关的东西的描述介绍，见第 7.3 节 “时间戳”

包列表的时间戳格式预设和精度可在浏览菜单选择，见第 3.5 节 “"File"菜单”

可用的预置格式如下:

- **Date and Time of Day: 1970-01-01 01:02:03.123456** 包捕捉的绝对日期和时间
- **Time of Day: 01:02:03.123456** 包捕捉的绝对时间
- **Seconds Since Beginning of Capture: 123.123456** 相对与文件开始捕捉的时间或第一个时间参考包的 到这个包之前的时间。(见第 6.10.1 节 “包参考时间”)
- **Seconds Since Previous Captured Packet: 1.123456** 相对前一个捕捉包的时间
- **Seconds Since Previous Displayed Packet: 1.123456** 相对前一个显示包的时间（过滤／显示）

可用精度(正如你所致的，数字是以10进制形式的)有：

- **Automatic** 使用载入文件格式具有的时间戳精度。(默认选项)
- **Seconds, Deciseconds, Centiseconds, Milliseconds, Microseconds or Nanoseconds** 强制使用你指定的精度。如果实际精度比你指定的低，会在后面自动追加0.如果实际精度比你指定的高。数据会被截尾。

精度距离：如果你有个时间戳，显示时使用：“Seconds Since Previous Packet”，：它的值可能是1.123456.默认会采用"Automatic"精度设置，也就是来自libpcap格式文件的固有精度(百万分之一秒)。如果你指定精度为秒，则显示为1，如果你使用。纳秒(nanoseconds),将会显示为1.123456000.

6.10.1. 包参考时间

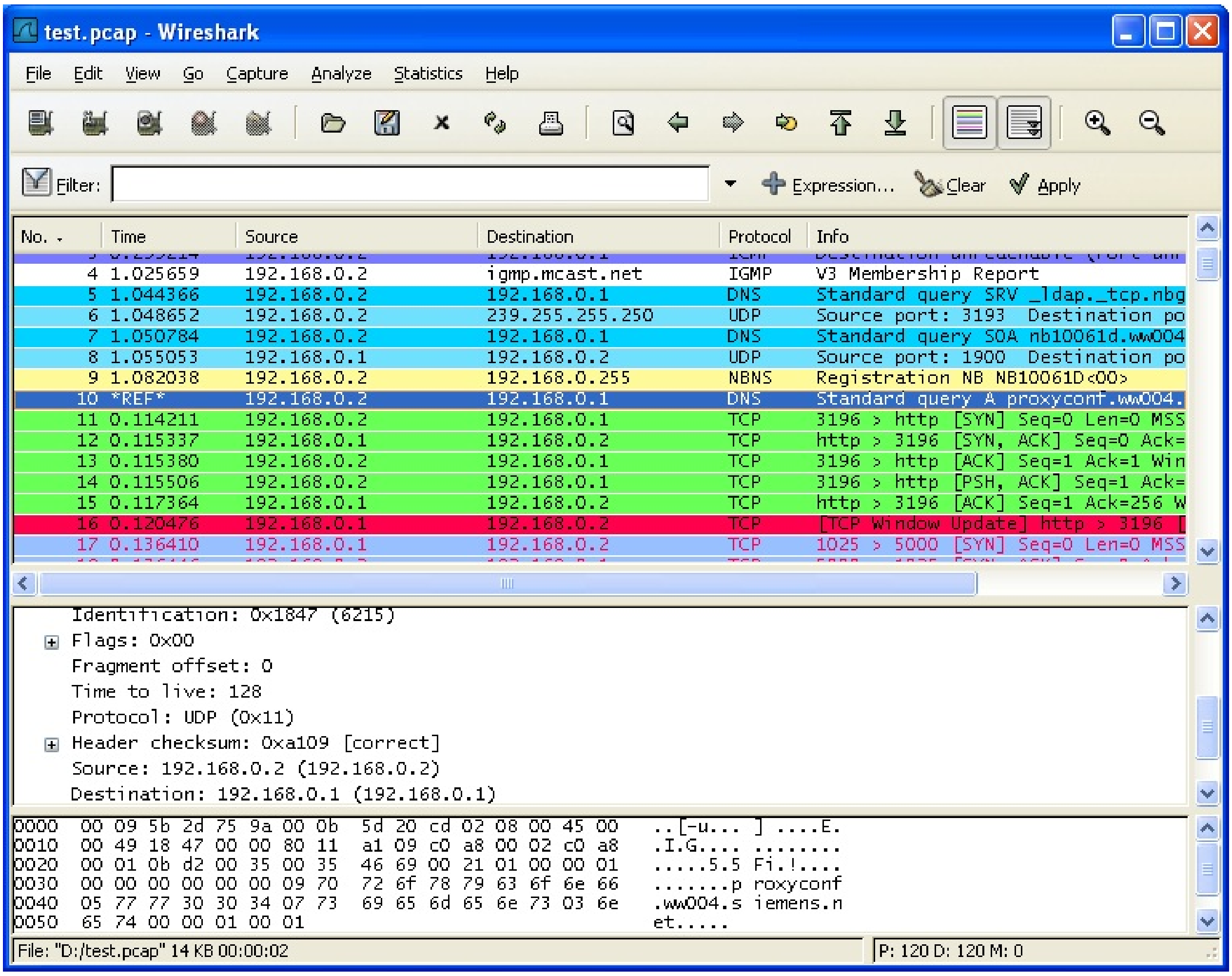
用户可以为包设置时间参考。时间参考是所有后续包的起算时间。如果你想知道到某一个特定包的时间间隔,会很有用。例如：开始一个新请求。可以在一个包里面设置多个参考时间。

> 警告
>
> 时间参考不能保存到包文件中，关闭文件后就会丢失。
>
> 注意
>
> 时间参考可能仅仅在时间格式为"Seconds Since Beginning of Capture"模式下比较有用。其他时间显示形式下可能要么是不能工作，要么是没作用。

要使用时间参考，选择Edit菜单下“Time Reference”项中的一个。详见第 3.6 节 “"Edit"菜单”，或者从包列表的右键弹出项选择。

- **Set Time Refernce(toggle)** 切换当前包时间参考状态开关
- **Find Next** 在包列表面板查找下一个时间参考包
- **Find Previous** 在包列表面板查找前一个时间参考包

图 **6.10.** 时间参考举例



作为时间参考的包，在time列会有*REF*字符串作为标记(见上图第10个包)。所有后续包都会用最后一个时间参考来显示时间。

第 7 章 高级

目录

7.1. 说明

在本节将介绍Wireshark的一些高级特性

7.2. "Follow TCP Stream"

如果你处理TCP协议，想要查看Tcp流中的应用层数据，"Following TCP streams"功能将会很有用。如果你项查看telnet流中的密码，或者你想尝试弄明白一个数据流。或者你仅仅只需要一个显示过滤来显示某个TCP流的包。这些都可以通过Wireshark的"Following TCP streams"功能来实现。

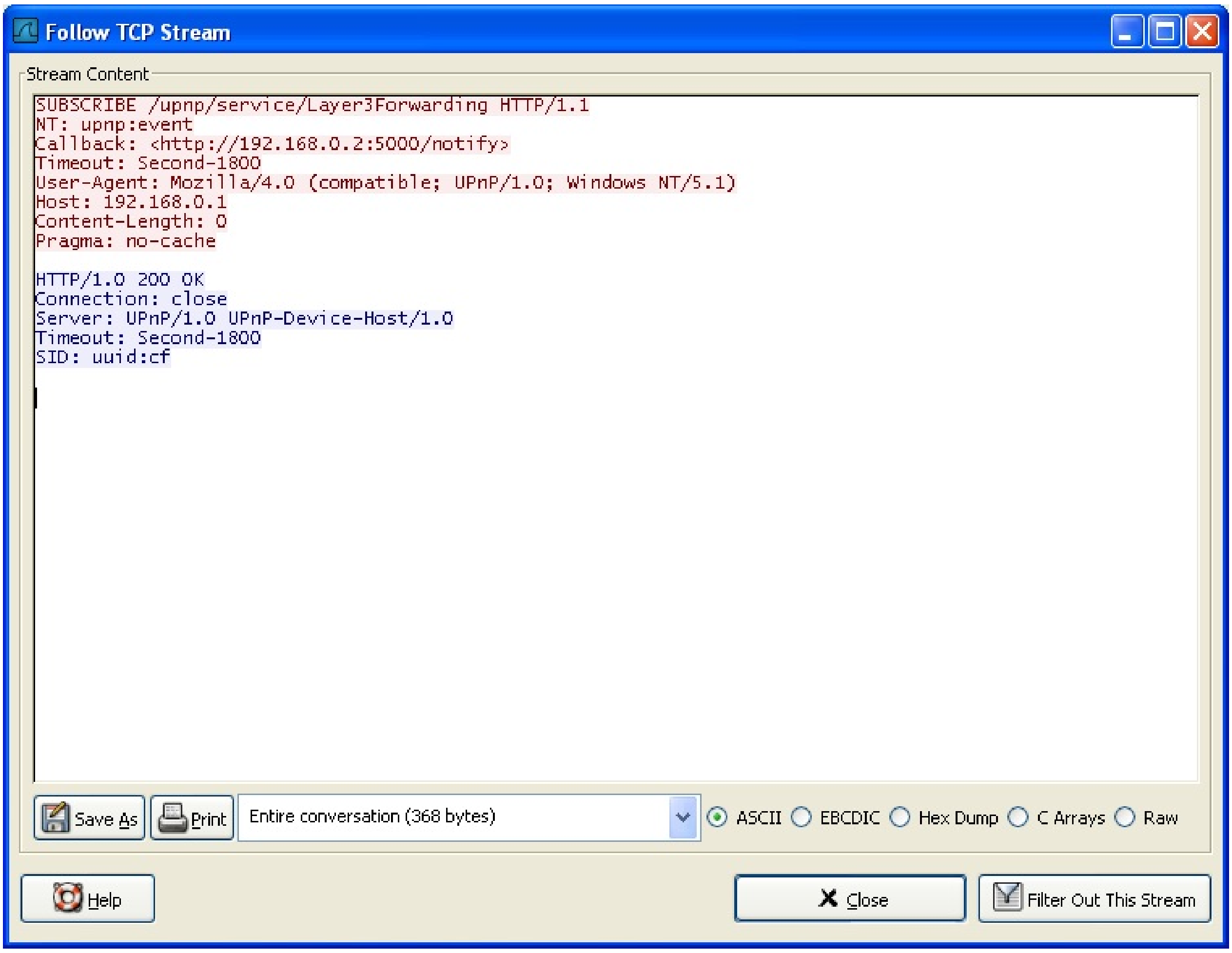
在包列表中选择一个你感兴趣的TCP包，然后选择Wireshark工具栏菜单的"Following TCP Streams"选项(或者使用包列表鼠标右键的上下文菜单)。然后，Wireshark就会创建合适的显示过滤器，并弹出一个对话框显示TCP流的所有数据。如图 7.1 “"Follow TCP Stream"对话框”所示

注意

值得注意的是：Follow Tcp Stream会装入一个显示过滤来选择你已经选择的Tcp流的所有包。

7.2.1. "Follow TCP Stream"对话框

图 **7.1. "Follow TCP Stream"对话框**



流的内容出现的顺序同他们在网络中出现的顺序一致。从A到B的通信标记为红色，从B到A的通信标记为蓝色。当然，如果你喜欢的话你可以从"Edit/Preferences"菜单项的"Colores"修改颜色。

非打印字符将会被显示为圆点。XXX - What about line wrapping (maximum line length) and CRNL conversions?

在捕捉过程中，TCP流不能实时更新。想得到最近的内容需要重新打开对话框。

你可以在此对话框执行如下操作：

1. **Save As** 以当前选择格式保存流数据。
2. **Print** 以当前选择格式打印流数据。
3. **Direction** 选择流的显示方向("Entire conversation", "data from A to B only" or "data from B to A only").
4. **Filter out this stream** 应用一个显示过滤，在显示中排除当前选择的TCP流。
5. **Close** 关闭当前对话框。移除对当前显示过滤的影响。

你可以用以下格式浏览流数据。

1. **AsCII**。在此视图下你可以以ASCII凡是查看数据。当然最适合基于ASCII的协议用，例如HTTP.
2. **EBCDIC**。For the big-iron freaks out there.（不知道这句是什么意思， EBCDIC 是IBM公司的字符二进制编码标准。）
3. **HEX Dump**. 允许你查看所有数据，可能会占用大量屏幕空间。适合显示二进制协议。
4. **C Arrays.** 允许你将流数据导入你自己的C语言程序。
5. **RAW**。 允许你载入原始数据到其他应用程序做进一步分析。显示类似与ASCII设置。但“save As”将会保存为二进制文件。

7.3. 时间戳

时间戳，时间戳的精度，等等是在让人感到困惑。本节将向你介绍介绍Wireshark处理时间戳时都发生了什么。

在包被捕捉时，每个包在进入时都被加上时间戳，这个时间戳将会保存在捕捉文件中，所以他们也可以在以后分析时使用。

那么说，时间戳是从哪里来的呢？是捕捉的时候产生的。Wireshark从 libpcap(WinPcap) libraray(库)中获得时间戳。而libpcap(winpcap)又是从操作系统内核获得的时间。如果捕捉数据是从捕捉文件载入的，很显然Wireshark从文件中获得时间戳数据。

7.3.1. Wireshark内置

Wireshak内置的格式使用的时间戳格式由日期(从1.1.1970开始)和时间（从凌晨起，纳秒(10亿分之一秒)为单位）组成。你可以调整Wireshark在包列表的时间戳显示方式。见第 3.7 节 “"View"菜单”的"Time Display Format"项。

当读取或写入捕捉文件时，Wireshark按需要在内置格式和其他捕捉文件格式间进行时间戳转换。

捕捉时，Wireshark使用libpcap(WinPcap)捕捉库（支持纳秒精度）。除非你在专用的捕捉硬件上进行捕捉，一般这样的精度已经足够了。

7.3.2. 捕捉文件格式

Wireshark支持的捕捉文件格式都带有时间戳。不同的捕捉文件格式的时间戳精度有很大不同，从秒"0"到纳秒 "0.123456789"都有。大多数格式捕捉文件存储的时间戳都是固定精度的，些捕捉文件格式甚至存储了时间戳精度本身（可能是出于方便）。

大多数被Wireshark(和或多其他工具)使用的libpcap捕捉文件格式都仅支持固定的百万分之一固定精度"0.123456"

> 注意
>
> 写入数据到一个实际支持精度比你写入数据精度低的格式文件中，可能会导致数据丢失。例如：如果你载入一个纳秒精度的捕捉文件，然后将其存储为libpcap文件(百万分之一秒精度)。Wireshark很明显会将时间精度调整为百万分之一秒。

7.3.3. 准确性

经常有人问："Wireshark的时间戳的准确性如何？"。实际上，Wireshark自身不会创建时间戳，而是通过其他的地方得到时间并显示他们。所以，准确性取决于你实用的捕捉系统(操作系统，性能。。。)。因此以上问题很难通过通常的途径回答。

> 注意
>
> 通常USB连接的网络适配器提供的精度非常差。入口的实际上“占用很长的时间和走很曲折的路”才能穿过USB数据线到系统内核。而数据包只有被系统内核处理过以后才会打上时间戳，这种时间戳机制将会导致准确性大大降低。
>
> 结论：如果你需要精确的时间戳，请不要使用USB连接的网卡！(笔者的注脚：有没有网卡在USB硬件上提前加上时间戳的？)[17]

side bar ceshi

[17] 译者注:前文提到，时间戳是Wireshark用库获取的时间加在包上的，不知为何有此一问。难道以后要识别硬件是否有时间戳功能。

7.4. 时区

当你在各地旅行时，会碰到时区的困扰。如果你从其他时区得到捕捉文件，时区问题会给你带来更大的困扰:-)

首先，下面有两个原因说明你为什么完全不需要考虑时区问题：

- 你仅对两个包的时间戳的差别有兴趣，你并不需要了解捕捉包的实际的日期和时间(通常是这样)。
- 很可能你不会得到不同与你所在时区的包文件，所以你基本上碰不到时区问题。例如：你的团队的所有都和你工作在一个时区。

表 **7.1.**

什么是时区**?**

人们希望时间和日升日落对应。早成应该是6点钟，天黑应该在20：00.这些时间又随着四季变化。如果地球上每个人使用同样的全局时间，将只有一小部分人的日落和时间对应，这将会导致混乱。

因此，人们将地球划分为不同的区域，每个区域都有一个本地时间对应本地的日升日落。

时区基于UTC(Coordinated Universal Time)或者Zulu 时间(军事和航空)。旧有的GTM(格林尼治时间)已不再使用，因为它有少许误差(与UTC相比达到0.9秒)。UTC起始时区等于0(位于格林威治，英格兰)，所有的时区和它的偏在在-12~+14小时之间！

例如：如果你住在柏林，你的时区将比UTC早1小时，所以你的时区应该是"+1"(与UTC时间比较的差别，以小时为单位)。柏林的3点和UTC的两点钟表示同一个时刻。

有些地区要加以注意，因为那里的时区不是用整小时的。(比如：新德里的时区是UTC+05:30)

更多相关信息见http://en.wikipedia.org/wiki/time_zone和http://en.wikipedia.org/wiki/Coordinated_Universal_Time。

表 **7.2.**

什么是时DST?

Daylight Saving Time(DST),又称为夏令时，目的是在夏天的几个月里“拯救”白天的时间(夏季白昼较长，如果按照传统的作息时间，比较可惜，不过我不认为)。为了达到这个目的，很多国家（但不是所有的）增加一个DST小时到UTC中。所以你还得加个小时(极少数地方甚至是2小时)的时差来计算你的时区。

不幸的是，DST并未在全世界范围内被启用。你可能同样注意到，北半球和南半球的夏令时是刚好相反的（比如：欧洲是夏季时，澳大利亚则是冬季）。

注意：不管DST怎样，UTC在全年都是一致的。

更多相关信息见http://en.wikipedia.org/wiki/Daylight_saving.

7.4.1. 正确设置你的计算机的时区

7.4.2. Wireshark和时区的关系

7.5. 合并包

7.5.1. 什么是合并包

网络协议经常需要传输较大的数据块，在传输时，底层协议可能不支持这样大的数据块(比如网络包大小的限制)，或者是像像TCP一样的流数据,TCP流完全不知道数据分块情况。(原文为:or is stream-based like TCP, which doesn't know data chunks at all.)

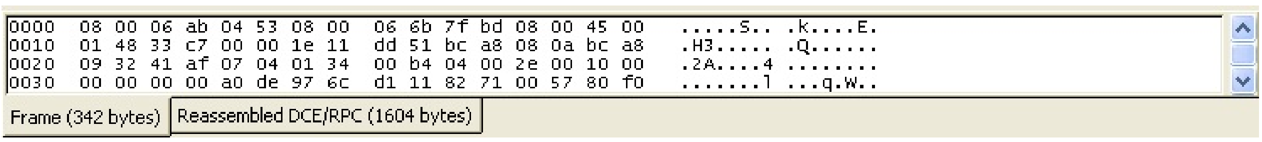
在这种情况下，网络协议必须确定数据块分段的边界，并(如果有必要)将数据块分割为多个包。很明显在接受端也需要有找到数据块分段边界的机制。

> 提示
>
> 在Wireshark里面，这个机制/方法被称为合并/**reasembling**，在特定协议的描述可能不尽相同(例如：desegmentation, defragmentation, ...)

7.5.2. 如何用Wireshark合并包

对那些可以被Wireshark识别的协议，Wireshark通常处理过程为：查找、解码、显示数据块。Wireshark会尝试查找数据块对应的包，在"Packet Bytes"面板的附加页面显示合并以后的数据。(关于“Packet Bytes”面板的详细介绍，见第 4.7 节 “"View"菜单”)

图 7.2. 带有合并包附加选项卡"Packet Bytes面板"



> 注意
>
> 合并可能发生在多个协议层，所以在"Packet Bytes"面板可能会见到多个附加页选项卡

> 注意
>
> 你会在数据块的最后一个包看到合并后的数据。

以HTTP Get应答为例：请求数据(例如一个HTML页面)返回时。Wireshark会显示一个16进制转储数据在"Packet Bytes"面板的一个名为"Uncompressed entity body"新选项卡。

默认情况下，首选项中合并功能被设置为允许。在2005年9月之前默认值是不允许。如果你的首选项是在200年9月之前设置的，你得确认一下，合并包选项的设置。合并包对分析网络包作用非常大。

允许和禁止合并包设置对协议来说还有两项要求。

1. 下层的协议(如：TCP)必须支持合并。通常协议支持合并与否都是通过协议的参数设置的。
2. 高层协议协议(如:HTTP)必须使用合并机制来合并分片的数据。这些也同样可以通过协议参数来允许或禁止。

在设置高层协议的时候tooltip会提醒你同样需要考虑低层的协议设置。

7.6. 名称解析

名字解析尝试将数字地址解析成适合人们阅读格式。有两种方法可以完成这项工作：通过系统/网络服务(例如获取主机名)和/或 Wireshark指定的赋值文件。关于通过赋值文件进行解析的详情，可以参见???

名字解析可以分协议层进行允许，禁止设置。

7.6.1. 名字解析的流弊

名字解析在使用Wireshark时有重要价值，甚至可以节约大量时间。不幸的是，名字解析也有它自己的缺点。

- 名字解析经常会不可用。服务器可能不知道需要被解析的名字，或者服务器不可用。又或者需要解析的名字在赋值配置文件中找不到。

- 名字解析并没有存储在捕捉文件或其他什么地方。因此你以后打开捕捉文件或者在其他设备上打开文件有可能发现名字解析不可用。每次打开捕捉文件可能会发现部分地址略微发生变化，也许仅仅是因为无法连接到名字解析服务器(之前还是可以连接的)
- **DNS**可能会增加额外的包到**Wireshark**中。你会在包文件中看到由Wireshark请求DNS服务生成的包进出你的机器。
- 解析名称被**Wireshark**缓存。这对设备性能有一定需求。但是，如果名字解析信息在wireshark运行时发生变化，wireshark不会注意到这个变化，因为它是从缓存进行解析的。如果这些信息在Wireshark运行时变化了，例如获取一个新DHCP租约，Wireshark不会注意到。(这些是针对DNS还是所有信息？有多少机器使用动态dns注册？)

> 提示
>
> 名字解析在包列表填入时已经完成。如果一个包填入包列表以后被解析，包列表的内容不会立即更改，相反解析结果会被缓存，你可以使用"View/Reload"重建包列表，来正确显示名字解析结果。但在捕捉过程中这样做没有效果。

7.6.2. 以太网名字解析(mac层)

尝试将MAC地址(e.g. 00:09:5b:01:02:03)解析为适合阅读的地址（"Human readable"）

ARP名字解析(系统服务) Wireshark会向操作系统发出请求，将以太网地址转换为对应的IP地址(e.g. 00:09:5b:01:02:03->192.168.0.1)

Ethernet codes(ethers file) 如果ARP解析错误，Wireshark会尝试将以太网地址解析为已知设备名。这种解析需要用户指定一个ethers文件为mac地址分配名称。(e.g. 00:09:5b:01:02:03 -> homerouter).

Ethernet manufacturer codes (manuf file) 如果ARP解析和ethers文件都无法成功解析，Wireshark会尝试转换mac地址的前三个字节为厂商名的缩写。mac地址的前三个字节是IEEE为各厂商分配的独立地址(通过前三个字节可以得出每个网络设备的供应商，当然这些也是可以被篡改的。，)(e.g. 00:09:5b:01:02:03 -> Netgear_01:02:03).

7.6.3. IP地址解析(网络层)

将IP地址(e.g. 216.239.37.99)转换为适合阅读的地址/"Human readable"

DNS/ADNS name resolution(system/library service) Wireshark会向操作系统（或ADSN library 地址-名称解析词典?）请求，将IP地址转换为相关联的主机名(e.g. 216.239.37.99 -> www.1.google.com).此时DNS服务正在同步请求DNS服务器，所以Wireshark会停止相应直到

DNS请求的响应返回。如果可能的话，你可以考虑使用ADNS library(这样可以避免等待网络相应。)

> 警告
>
> 如果名称解析服务器不可用，允许网络名称解析使Wireshark明显变慢，因为wireshark会等待名字解析结果直到超时。在这种情况你应该使用ADNS。

DNS vs. ADNS 这里是一个简短的对比：两个都是用来转换ip地址为其他易读的地址"Human readable"(域名)。通常DNS用gethostname()将地址转换为名称。通常首先是查询hosts文件(e.g. /etc/hosts,/windows/system32/drivers/etc/hosts)看能否找到匹配实体。如果找不到，会向指定的DNS服务器查询。

DNS和ADNS真正的区别在于等待DNS服务器名字解析。gethost()会一直等待知道名字被解析或者返回错误。如果DNS服务器不可用，可能会占用很长时间(好几秒)。ADNS服务会略微有点不同。它也同样向DNS服务器发出请求，但不会等待服务器应答。它会立即相应Wireshark。此时的地址（和后续地址）在ADNS得到结果前不会显示解析名称。如前文书中说道，解析结果被保存在缓存中，你需要使用"View/Reload"菜单更新这些字段来显示解析名称。

hosts name resolution(hosts file) 如果dns解析不成功，Wireshark会尝试使用用户提供的主机文件将IP地址转换为对应的主机名。(e.g. 216.239.37.99 -> www.google.com)

7.6.4. IPX名称解析(网络层)

ipxnet name resolution (ipxnets file) (笔者未作解释)

7.6.5. TCP/UDP端口名解析(传输层)

翻译TCP/UDP端口(e.g.80)为更加易读的玩意"human readable"[15]

TCP/UDP port conversion (system service) Wireshark会向操作系统发出请求，转换TCP/UDP端口为已知名称(e.g. 80->http)。

XXX - mention the role of the /etc/services file (but don't forget the files and folders section)!

[15] 应该是指将端口翻译为服务名

7.7. 校检和

很多协议使用校检和来验证数据的完整性/正确性。

提示

应用校检和在这里也被成为**redundancy check**(冗余校检？)

校检和是做什么的？

校验和是用来验证传输数据或存储数据的数据部分的正确性。一个检验和是数据部分进行摘要计算的出的数字。

网络数据在传输过程中经常会产生错误，例如数据错误，字节重复等。数据接收方可能。

正因为传输过程中会伴随错误，网络协议会经常使用校验和检测这些错误。发送方会对数据进行检验和计算，并将数据和检验和一起发送。接收方使用同样的方法计算数据部分的校验和，如果收到的校验和计算出来的校验和不匹配，就表示数据有错误。

有些校验和方法可以通过计算得出发生需要被修复错误的数据位置，并修复（简单的）错误。

如果那些错误无法修复，接收方会丢弃错误的数据包。根据协议的不同，数据丢失会仅仅被丢弃，也有可能发送端会根据数据丢失情况重传需要的数据包。

使用校验和可以大量减少传输错误数量。但任何检验和算法都无法确保100%检测到所有错误，依然有少量的错误会无法被检测到。

校验和的算法有很多，例如最经常被使用的检验和算法CRC32（循环冗余校验）。特的的网络协议选择的校验算法取决于希望网络媒介达到的出错率上限、错误检测的重要性，处理载入计算的性能，其他方面需要的性能。

关于检验和的更多信息可以参考：http://en.wikipedia.org/wiki/Checksum

7.7.1. Wireshark校检和验证

Wireshark会对很多协议进行检验和验证，如：TCP、IP。。。

它会和"normal receiver"做一样的计算.然后在包详情面板显示检验和字段的内容，e.g.: [correct], [invalid, must be 0x12345678] 以及其他类似的内容。

如果校验和验证选项被打开或正在进行校验和检测，合并包特性不会被执行。这是为了避免错误的的连接数据扰乱内部数据。

7.7.2. Checksum offloading

检验和计算可能由网络网络驱动，协议驱动，甚至是硬件完成。

例如：以太网传输硬件计算以太网循环容易校验，接受硬件验证这个校验。如果接受验证发现错误，Wireshark将不会接收到这个包，以太网硬件会直接丢弃这个包。

高层校验通常是由协议执行，并将完成后的包转交给硬件。

比较新的网络硬件可以执行一些高级功能，如IP检验和计算，这被成为checksum offloading。网络驱动不会计算校验和，只是简单将校验和字段留空或填入无效信息，交给硬件计算。

> 注意
>
> checksum offloading经常会导致混乱，因为网络包在检验和计算之前转交给Wireshark。Wireshark得到包的检验和字段是空的，必然会显示检验和错误，尽管这个包在从网络硬件发出的时候是带有校验和的。

Checksum offloading会引起混淆，让你屏幕上看到大量的[invalid]信息，引起你的反感。前面提到过，错误的检验和会导致包无法合并，更难进行包数据分析。

你可以采取两种方法避免Checksum offloading 问题

- 在驱动程序上关闭checksum offloading选项，如果可用的话。[16]
- 通过首选项关闭Wireshark上特定协议的校验和验证。

[16] 在Windows平台如果驱动支持，应该是计算机管理->设备管理器->网络适配器->对应网卡的属性-高级选项

第 8 章 统计

目录

8.1. 说明

Wireshark提供了多种多样的网络统计功能

包括，载入文件的基本信息(比如包的数量)，对指定协议的统计(例如，统计包文件内HTTPP请求和应答数)，等等。

- 一般统计
  - **Summary**：捕捉文件摘要
  - **Protocal Hierarchy:** 捕捉包的层次结构
  - **Endpoints** 例如：通讯发起，终止方的ip地址
  - **Conversations** 例如：两个指定IP之间的通信
  - **IO Graphs** 包数目随时间变化的曲线图。
- 指定协议统计
  - **Service Response Time** 从发起请求到相应请求的服务间隔时间。

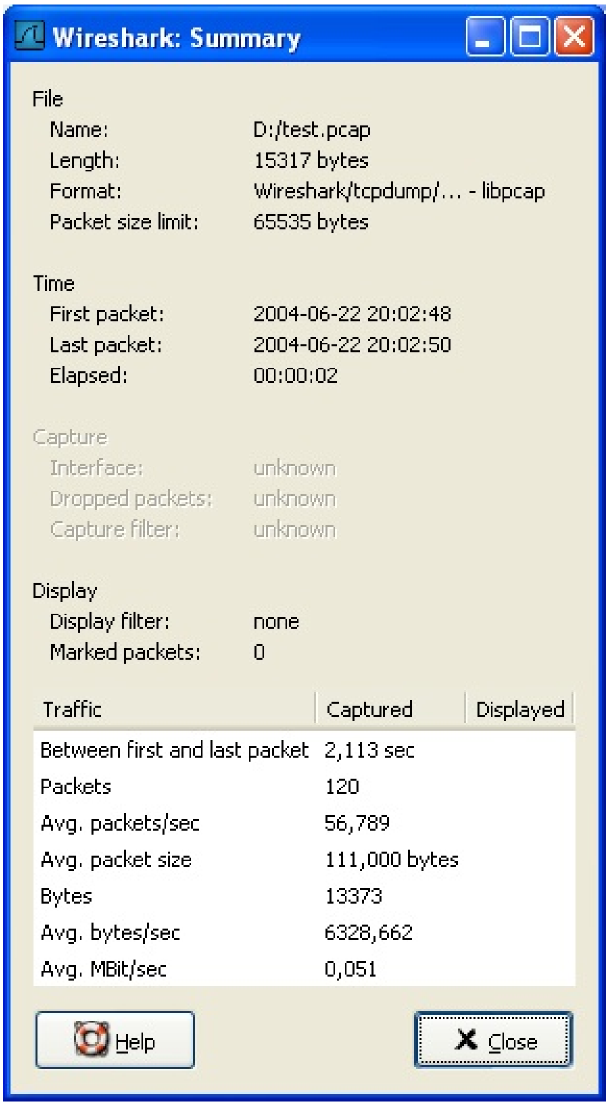
- **Various other** 协议特有的统计

> 注意
>
> 协议特定的统计，需要有特定协议的细节了解。除非你对那个协议非常熟悉，统计结果不是那么那么容易理解的。

8.2. 摘要窗口

当前捕捉文件的一般信息

图 **8.1. "Summary"** 窗口



File

捕捉文件的一般的信息

Time

第一个包和最后一个包的时间戳

Capture

包捕捉完成时的一些信息(仅当包数据已经从网络捕捉，还没有从文件载入)

Display

与显示有关的信息

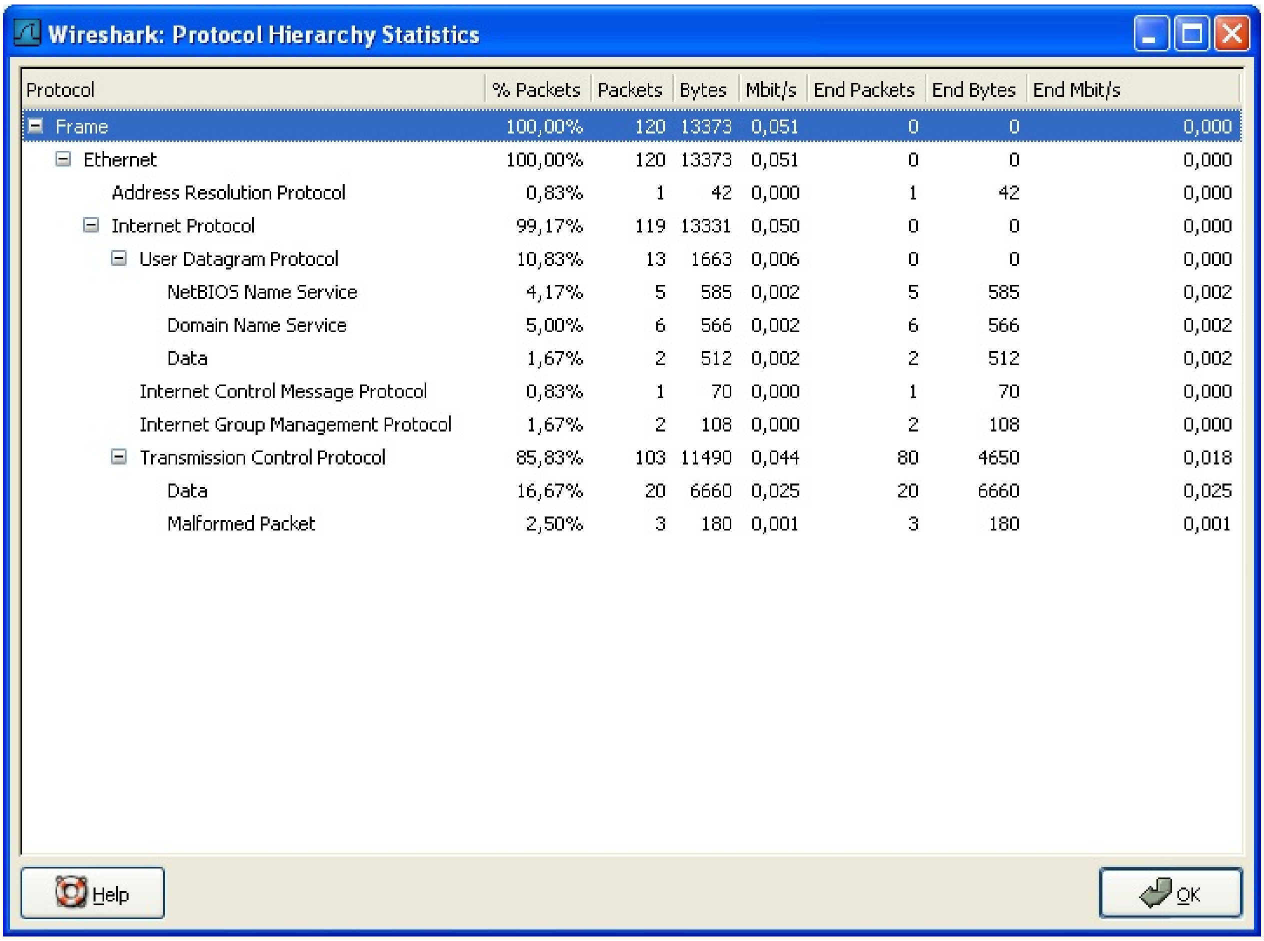
Traffic

网络传输的相关统计，如果设置了显示过滤，你会看到两列。**Captured**列显示过滤前的信息，**Displayed**列显示过滤后对应的信息。

8.3. "Protocol Hierarchy"窗口

显示捕捉包的分层信息

图 **8.2. "Protocol Hierarchy"** 窗口



Protocol	% Packets	Packets	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100,00%	120	13373	0,051	0	0	0,000
Ethernet	100,00%	120	13373	0,051	0	0	0,000
Address Resolution Protocol	0,83%	1	42	0,000	1	42	0,000
Internet Protocol	99,17%	119	13331	0,050	0	0	0,000
User Datagram Protocol	10,83%	13	1663	0,006	0	0	0,000
NetBIOS Name Service	4,17%	5	585	0,002	5	585	0,002
Domain Name Service	5,00%	6	566	0,002	6	566	0,002
Data	1,67%	2	512	0,002	2	512	0,002
Internet Control Message Protocol	0,83%	1	70	0,000	1	70	0,000
Internet Group Management Protocol	1,67%	2	108	0,000	2	108	0,000
Transmission Control Protocol	85,83%	103	11490	0,044	80	4650	0,018
Data	16,67%	20	6660	0,025	20	6660	0,025
Malformed Packet	2,50%	3	180	0,001	3	180	0,001

这个窗口现实的是捕捉文件包含的所有协议的树状分支。你可以展开或折叠分支，通过点击+/-图标。默认情况下，所有分支都是展开的。

每行包含一个协议层次的统计值

每列代表的意思

Protocol

协议名称

%Packets

含有该协议的包数目在捕捉文件所有包所占的比例

Packet

含有该协议的包的数目

Bytes

含有该协议的字符数

MBit/s

该协议的带宽，相对捕捉时间

End Packets

End Bytes

End MBit/s

注意

包通常会包含许多协议，有很多协议会在每个包中被统计。例如：截屏中包括99.17%的IP，85.83%的TCP协议(它们的和超过了100%)

注意

包的协议组成部分可以不包含高层协议，高层协议包统计百分比和可能并不等于100%，例如:截屏中TCP占85.83%，但是上层协议(HTTP...)却比85%更少。这可能是因为TCP协议，例如：TCP ACK 包不会被统计到高层协议。

注意

一个单独的包可以包含相同的协议不止一次，这种情况下，协议会被计数超过一次。例如某些通道配置的协议，IP层会出现两次。(通道封装的内容包括ip层，传输时将封装过在用IP封装一次)

8.4. "Endpoints"

端点不着的统计

提示

如果在其他网络工具工具中看到被称为**Hostlist/**主机列表的东西，在这里就是Endpoint了。

8.4.1. 什么是Endpoint?

一个网络端点是在特定的协议层的通信的逻辑端点。Wireshark端点统计会将列出下列端点:

Ethernet

以太网端点显示的是以太网MAC地址

Fibre Channel

笔者未描述

FDDI

FDDI端点是FDDI MAC地址

IPV4

IP端点是IP地址

IPX

笔者未介绍

TCP

TCP端点由IP地址和TCP端口组成，同样的IP地址加上不同的端口号，表示的是不同的TCP端点

Token Ring

Token Ring(令牌环)端点是Token Ring MAC地址

UDP

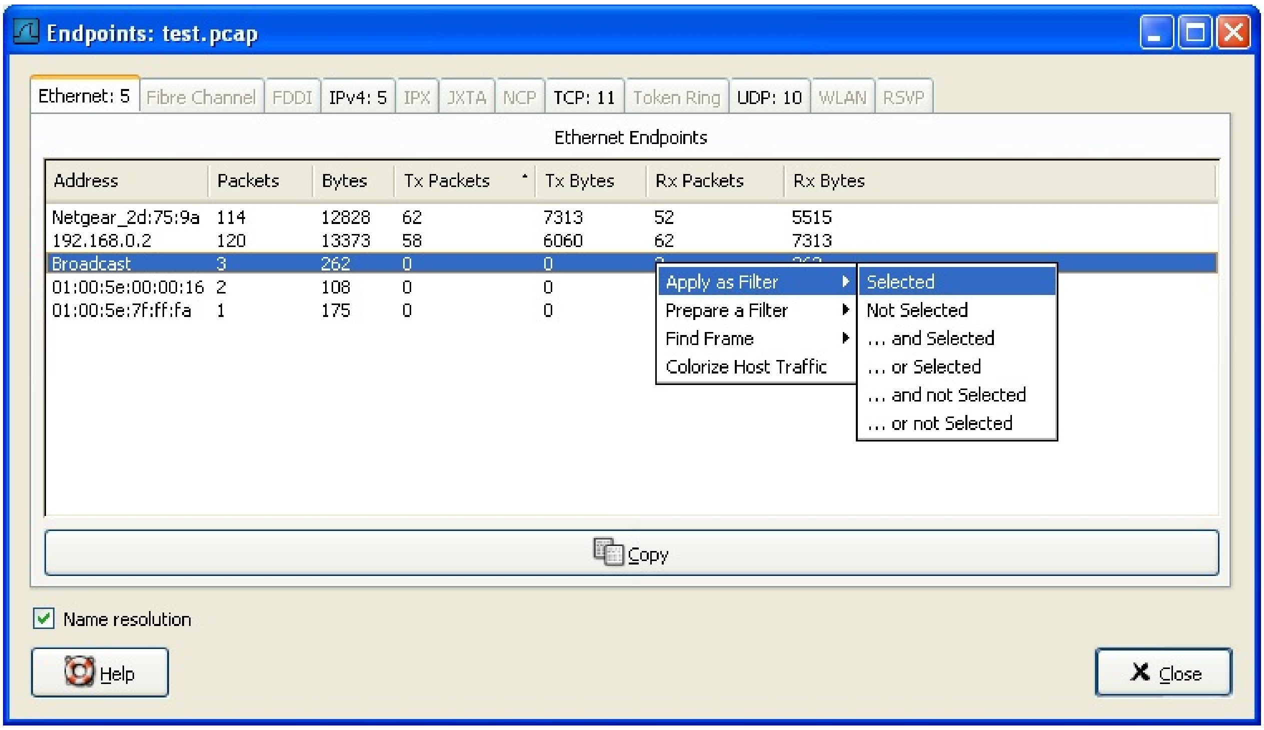
UDP端点是由IP地址和UDP端口组成，不同的UDP地址用同一个IP地址表示不同的UDP端点

> Broadcast / multicast endpoints（广播/多播端点）
>
> 广播/多播通信会用额外的端点单独显示。当然，这些端点都是虚拟端点，真实的通信会被所有(多播的一部分)列出的单播端点接收。

8.4.2. "Endpoints"窗口

该窗口显示端点捕捉的统计信息

图 8.3. "Endpoints"窗口



在该窗口中，每个支持的协议，都显示为一个选项卡。选项标签显示被捕捉端点数目(例如："Ethernet :5"表示有5个ethenet 端点被捕捉到)。如果某个协议没有端点被捕捉到，选项标签显示为灰色(尽管可以查看选项卡对应的页面).

列表中每行显示单个端点的统计信息。

Name resolution 如果选中该选项，将会对指定的协议层进行名字解析(当前选中的Ethernet endpoint 页面是MAC层)。你可能注意到，第一行将前三个字节解析为"Netgear",第二行地址被解析为IP地址(通过arp协议解析)，第三行解析为广播地址(未解析时mac地址为:ff:ff:ff:ff:ff:ff)，最后两行的MAC地址未被解析。



提示

该窗口可能会频繁那更新内容，在你进行实时捕捉之前打开了它(或者在这期间打开了它)，也依然有用。

8.4.3. 特定协议的"Endpoint List"窗口

Before the combined window described above was available, each of its pages were shown as separate windows. Even though the combined window is much more convenient to use, these separate windows are still available. The main reason is, they might process faster for very large capture files. However, as the functionality is exactly the same as in the combined window, they won't be discussed in detail here.

8.5. 会话/conversations

已经捕捉的会话统计

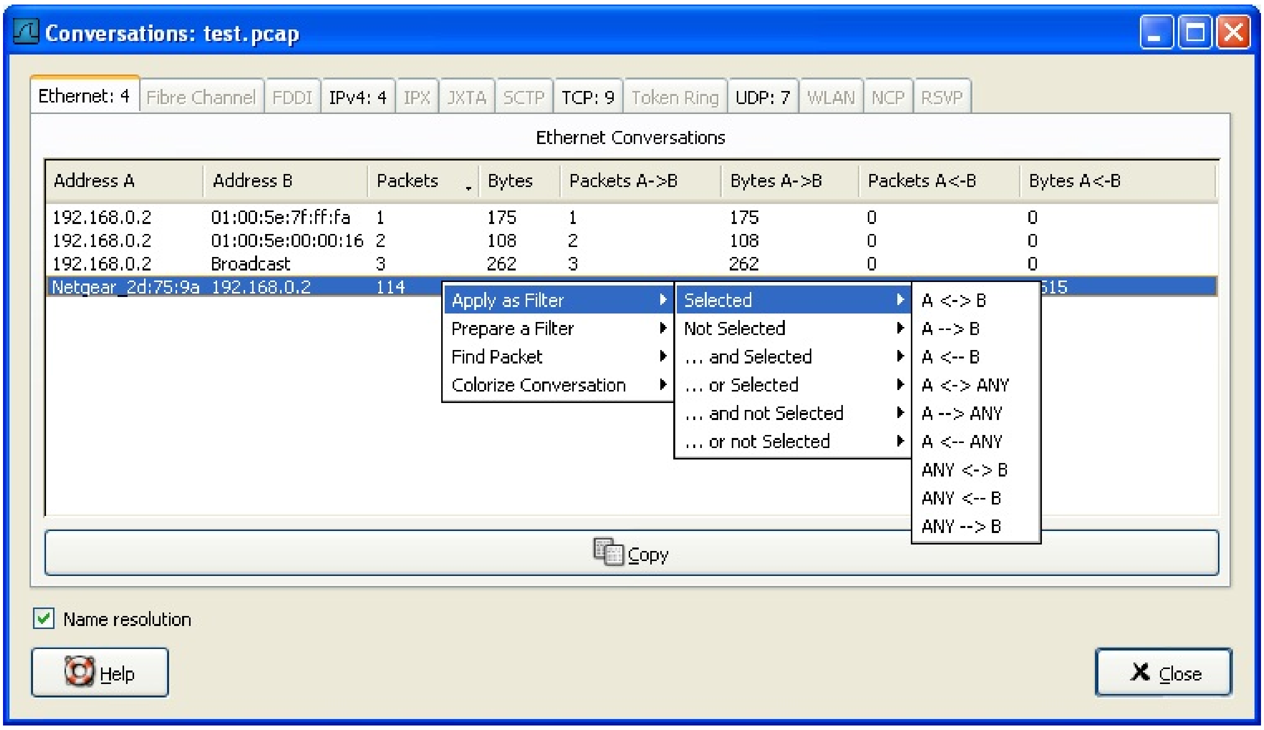
8.5.1. 什么是会话/conversation?

一个网络会话，指的是两个特定端点之间发生的通信。例如，一个IP会话是两个IP地址间的所有通信。

8.5.2. "Conversations/会话" window

除了列表内容之外，会话窗口和端点窗口基本一样，见第 8.4.2 节 “"Endpoints"窗口”

图 8.4. "Conversations"对话框



8.5.3. 协议指定“Conversation List/会话列表”窗口

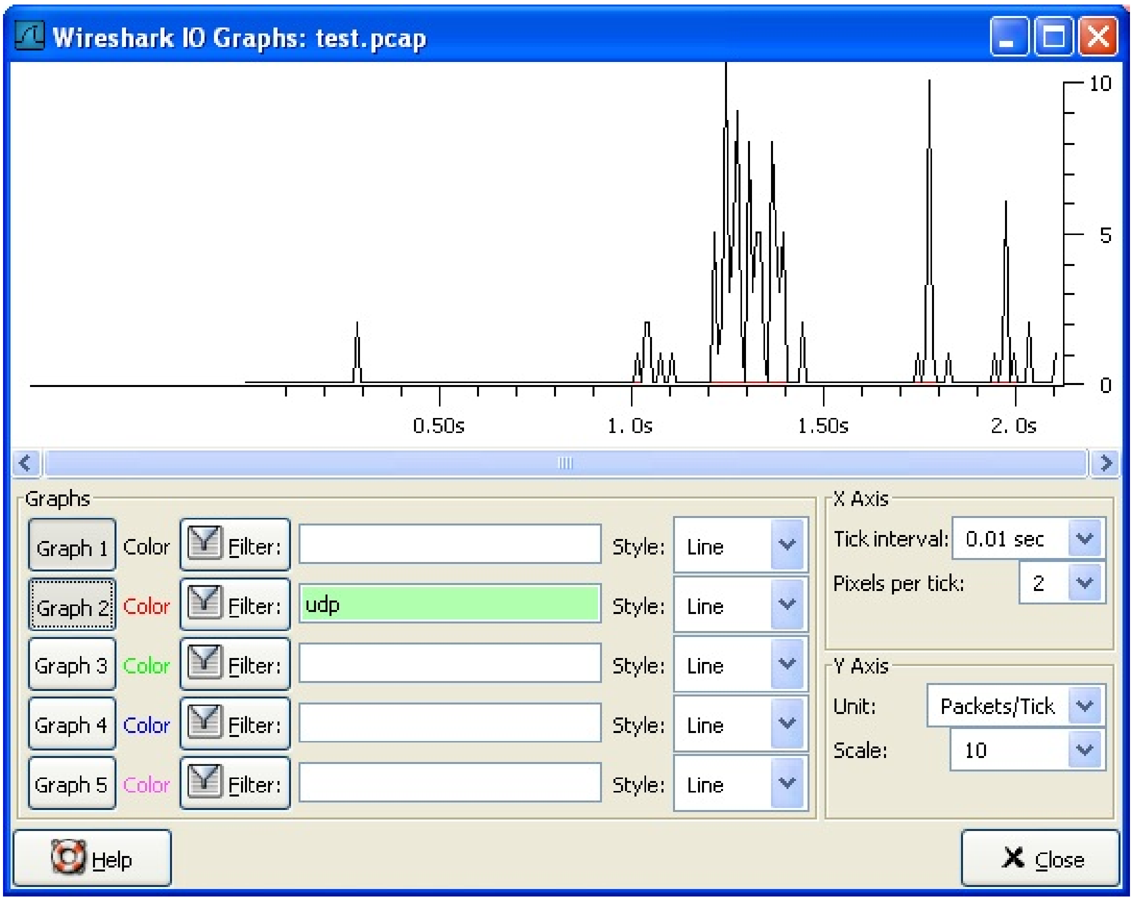
Before the combined window described above was available, each of its pages were shown as separate windows. Even though the combined window is much more convenient to use, these separate windows are still available. The main reason is, they might process faster for very large capture files. However, as the functionality is exactly the same as in the combined window, they won't be discussed in detail here.

8.6. "IO Graphs"窗口

用户可配置的捕捉网络数据图形。

你可以设置五种不同颜色的图形

图 **8.5. "IO Graphs" 窗口**



用户可以对一下内容进行设置。

Graphs

- **Graph 1-5:** 开启1-5图表(默认仅开启graph 1)
- **Color:** 图表的颜色(不可修改)
- **Filter:**指定显示过滤器(only the packets that pass this filter will be taken into account for that graph)
- **Style:**图表样式(Line/Impulse/FBar)

X Axis

- **Tick interval** 设置X轴的每格代表的时间(10/1/0.1/0.01/0.001 seconds))
- **Pixels per tick** 设置X轴每格占用像素 10/5/2/1 px

Y Axis

- **Unit y**轴的单位(Packets/Tick, Bytes/Tick, Bits/Tick, Advanced...)
- **Ssale Y**轴单位的刻度(10,20,50,100,200,500,...)

XXX - describe the Advanced feature.

8.7. 服务相应时间

服务响应时间是发送请求到产生应答之间的时间间隔。响应时间在很多协议中可用。

服务相应时间统计，在以下协议中可用

- DCE-RPC
- Fibre Channel
- H.225 RAS
- LDAP
- MGCP
- ONC-RPC
- SMB

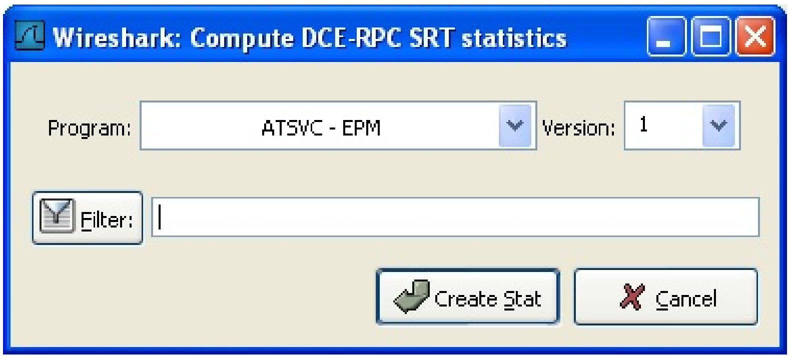
后面将会以DCE-RPC为例介绍响应时间。

> 注意
>
> 其他服务相应时间在Windows平台下都是相同的处理方法(或者仅仅轻微不同)

8.7.1. "Service Response Time DCE-RPC"窗口

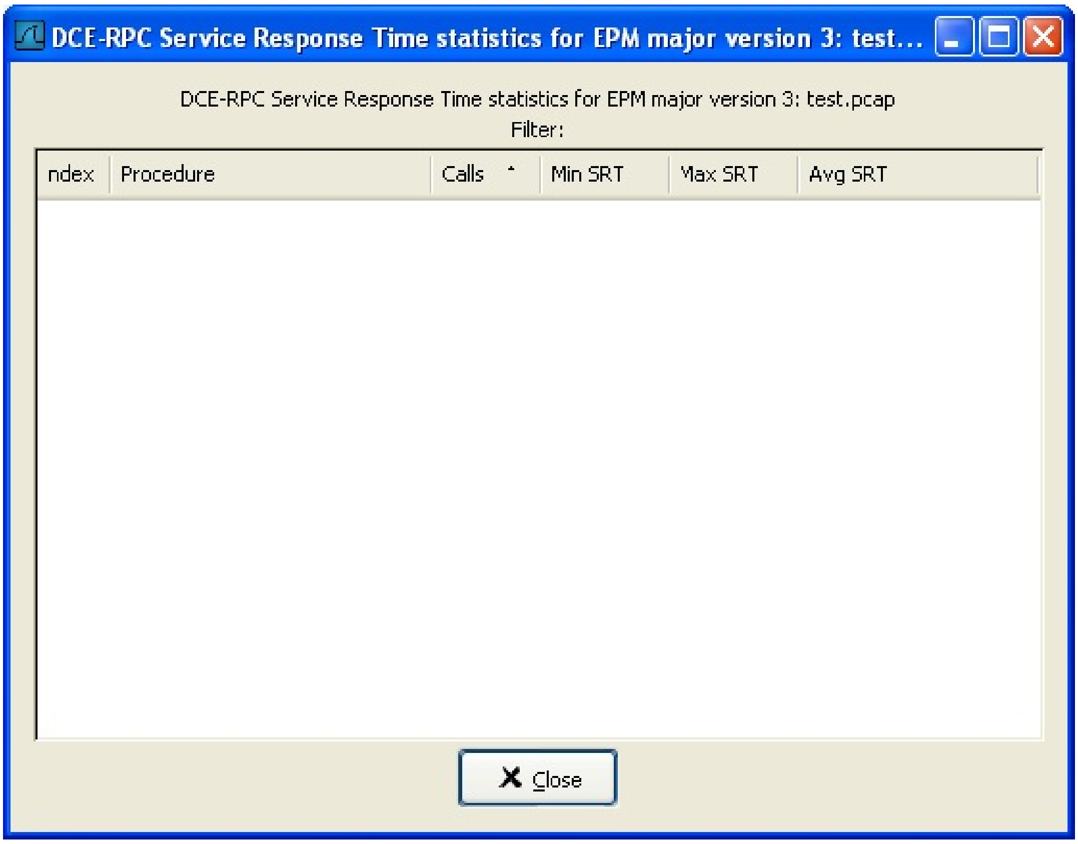
DCE-RPC的服务相应时间是在请求发起到相应请求的时间间隔

图 **8.6. "Compute DCE-RPC statistics"窗口**



你可以设置显示过滤，减少用于统计的包的数量

图 8.7. The "DCE-RPC Statistic for ..." 窗口



Each row corresponds to a method of the interface selected (so the EPM interface in version 3 has 7 methods). For each method the number of calls, and the statistics of the SRT time is calculated.

8.8. 协议指定统计窗口

The protocol specific statistics windows display detailed information of specific protocols and might be described in a later version of this document. Some of these statistics are described at the http://wiki.wireshark.org/Statistics pages.

第 9 章 个性化Wireshark

目录

> 写在前面
>
> 本章自9.6节起的内容在译者的0.99.5版Wireshark中都未曾见到对应的功能。

9.1. 说明

Wireshark默认行为通常可以很好地吻合你的习惯，当你十分熟悉Wireshark的时候，你可以对Wireshark进行个性化设置以更好地适合你的需要。在本章我们将介绍:

- 如何从命令启动Wireshark
- 如何将包列表色值化(以颜色区分不同的包)
- 如何控制包解析
- 如何使用多种多样的首选项设置

9.2. 从命令行启动Wireshark

Wireshark支持从命令行启动，同样也可以从大多数窗口管理软件启动。这节我们看看如何从命令行启动。

Wireshark支持丰富的命令行参数。要想看看都有那些参数，在命令行键入**Wireshark -h**就会显示帮助信息（以及其他相关的）。详细参数列表见例 9.1 “Wireshark帮助信息”

例 **9.1. Wireshark**帮助信息

```
Version 0.99.0
Copyright 1998-2006 Gerald Combs <gerald@wireshark.org> and contributors.

Compiled with GTK+ 2.6.9, with GLib 2.6.6, with WinPcap (version unknown),
with libz 1.2.3, with libpcre 6.4, with Net-SNMP 5.2.2, with ADNS, with Lua 5.1.

Running with WinPcap version 3.1 (packet.dll version 3, 1, 0, 27), based on
libpcap version 0.9[.x] on Windows XP Service Pack 2, build 2600.

wireshark [ -vh ] [ -DklLnpQS ] [ -a <capture autostop condition> ] ...
        [ -b <capture ring buffer option> ] ...
        [ -B <capture buffer size> ]
        [ -c <capture packet count> ] [ -f <capture filter> ]
        [ -g <packet number> ] [ -i <capture interface> ] [ -m <font> ]
        [ -N <name resolving flags> ] [ -o <preference/recent setting> ] ...
        [ -r <infile> ] [ -R <read (display) filter> ] [ -s <capture snaplen> ]
        [ -t <time stamp format> ] [ -w <savefile> ] [ -y <capture link type> ]
        [ -X <eXtension option> ] [ -z <statistics> ] [ <infile> ]
```

我们随后将对每个选项进行介绍

首先需要注意的是，**Wireshark**命令会启动Wireshark。不管怎样，你可以在启动时追加许多参数(如果你喜欢)。他们的作用如下(按字母顺序)：

笔者注：按字母顺序是不是一个好主意？按任务顺序会不会更好点？

-a <capture autostop condition>

设置一个标准用来指定Wireshark什么时候停止捕捉文件。标准的格式为 test:value,test值为下面中的一个。

duration:value

当捕捉持续描述超过Value值，停止写入捕捉文件。

filesize:value

当捕捉文件大小达到Value值kilobytes(kilobytes表示1000bytes,而不是1024 bytes)，停止写入捕捉文件。如果该选项和-b选项同时使用，Wireshark在达到指定文件大小时会停止写入当前捕捉文件，并切换到下一个文件。

files:value

当文件数达到Value值时停止写入捕捉文件

-b <capture ring buffer option>

如果指定捕捉文件最大尺寸，因为Wireshark运行在"ring buffer"模式，被指定了文件数。在"ring buffer"模式下，Wireshark 会写到多个捕捉文件。它们的名字由文件数和创建日期，时间决定。

当第一个捕捉文件被写满，Wireshark会跳转到下一个文件写入，直到写满最后一个文件，此时Wireshark会丢弃第一个文件的数据(除非将files设置为0，如果设置为0，将没有文件数限制)，将数据写入该文件。

如果duration选项被指定，当捕捉持续时间达到指定值的秒数，Wireshark同样会切换到下个文件，即使文件未被写满。

duration:value

当捕捉持续描述超过Value值，即使文件未被写满，也会切换到下个文件继续写入。

filesize:value

当文件大小达到value值kilobytes时(kelobyte表示1000bytes,而不是1024bytes)，切换到下一个文件。

files:value

当文件数达到value值时，从第一个文件重新开始写入。

-B <capture buffer size (Win32 only)>

仅适合Win32:设置文件缓冲大小(单位是MB,默认是1MB).被捕捉驱动用来缓冲包数据，直到达到缓冲大小才写入磁盘。如果捕捉时碰到丢包现象，可以尝试增大它的大小。

-c <capture packet count>

实时捕捉中指定捕捉包的最大数目，它通常在连接词-k选项中使用。

-D

打印可以被Wireshark用于捕捉的接口列表。每个接口都有一个编号和名称(可能紧跟在接口描述之后？)会被打印，接口名或接口编号可以提供给-i参数来指定进行捕捉的接口(这里打印应该是说在屏幕上打印)。

在那些没有命令可以显示列表的平台(例如Windows,或者缺少**ifconfig -a**命令的UNIX平台)这个命令很有用;接口编号在Windows 2000及后续平台的接口名称通常是一些复杂字符串，这时使用接口编号会更方便点。

注意，"可以被Wireshark用于捕捉"意思是说：Wireshark可以打开那个设备进行实时捕捉；如果在你的平台进行网络捕捉需要使用有特殊权限的帐号(例如root，Windows下的Administrators组)，在没有这些权限的账户下添加-D不会显示任何接口。参数

-f <capture filter>

设置捕捉时的内置过滤表达式

-g <packet number>

在使用-r参数读取捕捉文件以后，使用该参数跳转到指定编号的包。

-h

-h选项请求Wireshark打印该版本的命令使用方法(前面显示的)，然后退出。

-i <capture interface>

设置用于进行捕捉的接口或管道。

网络接口名称必须匹配**Wireshark -D**中的一个；也可以使用**Wireshark -D**显示的编号，如果你使用UNIX,**netstat -i**或者**ifconfig -a**获得的接口名也可以被使用。但不是所有的UNIX平台都支持**-a**,**ifconfig**参数。

如果未指定参数，Wireshark会搜索接口列表，选择第一个非环回接口进行捕捉，如果没有非环回接口，会选择第一个环回接口。如果没有接口，wireshark会报告错误，不执行捕捉操作。

管道名即可以是FIFO(已命名管道)，也可以使用"-"读取标准输入。从管道读取的数据必须是标准的libpcap格式。

-k

-k选项指定Wireshark立即开始捕捉。这个选项需要和-i参数配合使用来指定捕捉产生在哪个接口的包。

-l

打开自动滚屏选项，在捕捉时有新数据进入，会自动翻动"Packet list"面板（同-S参数一样）。

-m <font>

设置显示时的字体（编者认为应该添加字体范例）

-n

显示网络对象名字解析(例如TCP,UDP端口名，主机名)。

-N <name resolving flags>

对特定类型的地址和端口号打开名字解析功能；该参数是一个字符串，使用m可以开启MAC地址解析，n开启网络地址解析，t开启传输层端口号解析。这些字符串在-n和-N参数同时存在时优先级高于-n，字母C开启同时(异步)DNS查询。

-o <preference/recent settings>

设置首选项或当前值，覆盖默认值或其他从Preference/recent file读取的参数、文件。该参数的值是一个字符串，形式为 prefname:value,prefnmae是首选项的选项名称(出现在preference/recent file上的名称)。value是首选项参数对应的值。多个**-o <preference settings>** 可以使用在单独命中中。

设置单独首选项的例子：

wireshark -o mgcp.display_dissect_tree:TRUE

设置多个首选项参数的例子：

```
wireshark -o mgcp.display_dissect_tree:TRUE -o mgcp.udp.callagent_port:2627
```

> 提示
>
> 在???可以看到所有可用的首选项列表。

-p

不将接口设置为杂收模式。注意可能因为某些原因依然出于杂收模式；这样，-p不能确定接口是否仅捕捉自己发送或接受的包以及到该地址的广播包，多播包

-Q

禁止Wireshark在捕捉完成时退出。它可以和-c选项一起使用。他们必须在出现在-i -w连接词中。

-r <infile>

指定要读取显示的文件名。捕捉文件必须是Wireshark支持的格式。

-R <read(display) filter>

指定在文件读取后应用的过滤。过滤语法使用的是显示过滤的语法，参见第 6.3 节 “浏览时过滤包”，不匹配的包不会被显示。

-s <capture snaplen>

设置捕捉包时的快照长度。Wireshark届时仅捕捉每个包<snaplen>字节的数据。

-S

Wireshark在捕捉数据后立即显示它们，通过在一个进程捕捉数据，另一个进程显示数据。这和捕捉选项对话框中的"Update list of packets in real time/实时显示数据"功能相同。

-t <time stamp format>

设置显示时间戳格式。可用的格式有

- **r** 相对的，设置所有包时间戳显示为相对于第一个包的时间。
- **a** absolute,设置所有包显示为绝对时间。
- **ad** 绝对日期，设置所有包显示为绝对日期时间。
- **d** delta 设置时间戳显示为相对于前一个包的时间
- **e** epoch 设置时间戳显示为从epoch起的妙数(1970年1月1日 00:00:00起)

-v

请求Wireshark打印出版本信息，然后退出

-w <savefile>

在保存文件时以savefile所填的字符为文件名。

-y <capture link type>

如果捕捉时带有-k参数，-y将指定捕捉包中数据链接类型。The values reported by -L are the values that can be used.

-X <eXtension option>

设置一个选项传送给TShark 模块。eXtension 选项使用extension_key:值形式，extension_key:可以是：

lua_script:lua_script_filename,它告诉Wireshark载入指定的脚本。默认脚本是Lua scripts.

-z <statistics-string>

得到Wireshark的多种类型的统计信息，显示结果在实时更新的窗口。笔者注：在此处增加更多的细节

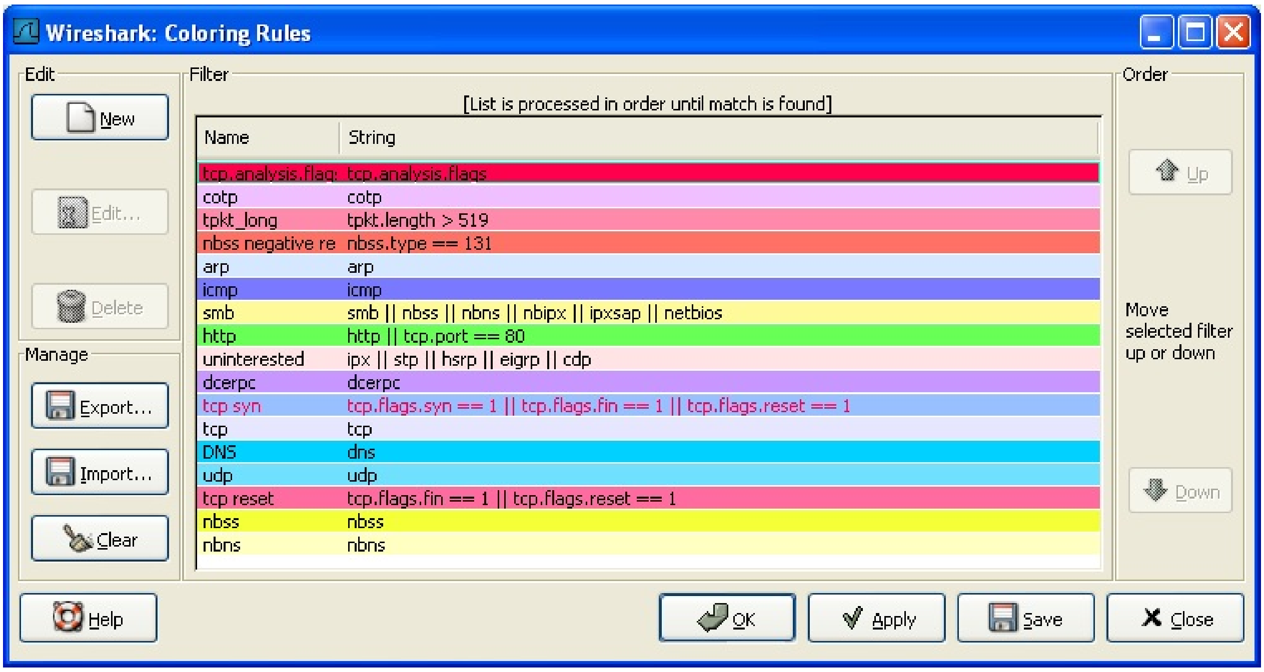
9.3. 包色彩显示设置

Packet colorization(按色彩显示包)是Wireshark一个非常有用的特性。你可以设置Wireshark通过过滤器将包按颜色设置。可以将你感兴趣的包通过颜色强调显示。

> 提示
>
> 你可以在http://wiki.wireshark.org/ColoringRules的**Wireshark Wiki Coloring Rules page**找到颜色规则的举例。

想要按色彩显示包，选择View菜单的“Coloring Rules...”菜单项，将会弹出"Coloring Rules"对话框，如图 9.1 “"Coloring Rules"对话框”所示

图 9.1. "Coloring Rules"对话框



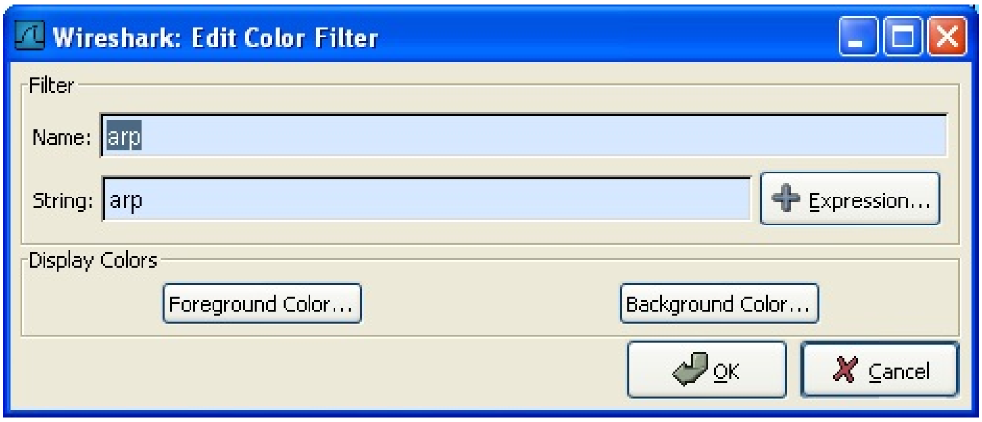
启动Coloring Rules 对话框以后，有许多按钮可以使用，当然这取决于是否已经装入颜色过滤器(碰到once sth,you have a lot of 之类的句子就觉得特别tmd的恶心。)

注意

在对色彩规则进行排序（然后运用时）需要注意：他们是按自上而下的顺序应用的。因此，特定的协议应该排在一般的协议的前面(高层协议应该排在底层协议之前)。例如：如果你将UDP协议排在DNS之前，那么DNS颜色规则就不会被应用(因为DNS使用UDP协议，UDP色彩规则首先被匹配。译者注：这里有点像netscreen防火墙规则，从上而下匹配，匹配了第一个规则以后就不会询问后续规则了。)

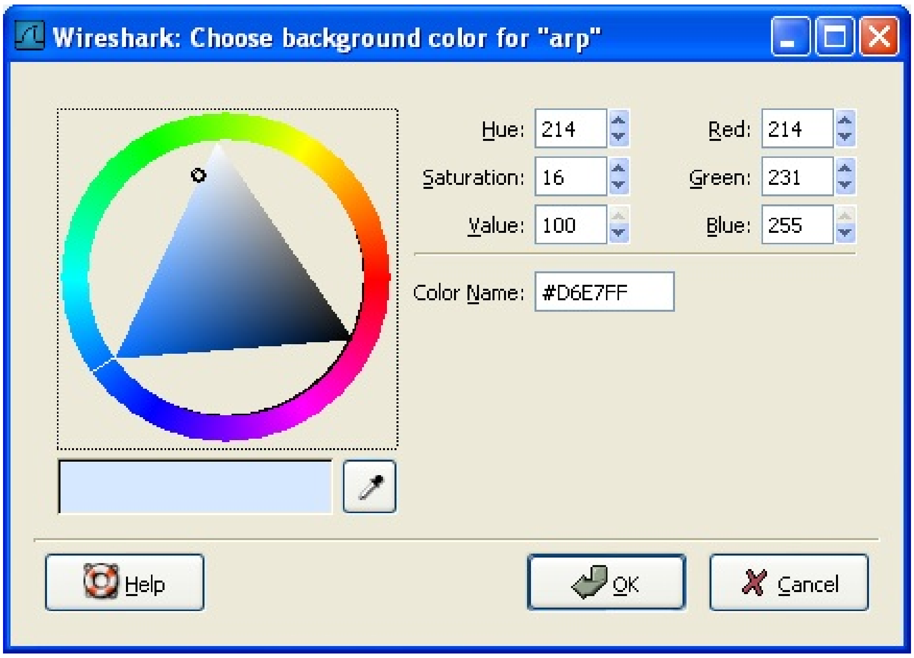
如果你第一次使用色彩规则，点击“NEW”按钮打开色彩过滤编辑对话框，如???所示：

图 9.2. "Edit Color Filter"



在编辑色彩对话框，输入颜色过滤器名称，然后在String输入框输入过滤字符串。???显示的是arp,arp表示过滤器名为arp,string填的arp表示选择的协议类型是arp。输入这些值以后，你可以选择前景色和配景色匹配这个过滤表达式。点击 **Foreground color...** /前景色或者**Background color...**/背景色按钮就会弹出**Choose foreground/background color for protocol**对话框(见图 9.3 “"Choose color"对话框”)，进行前景色、背景色设置了。

图 **9.3. "Choose color"**对话框



选择你需要的颜色，点击OK

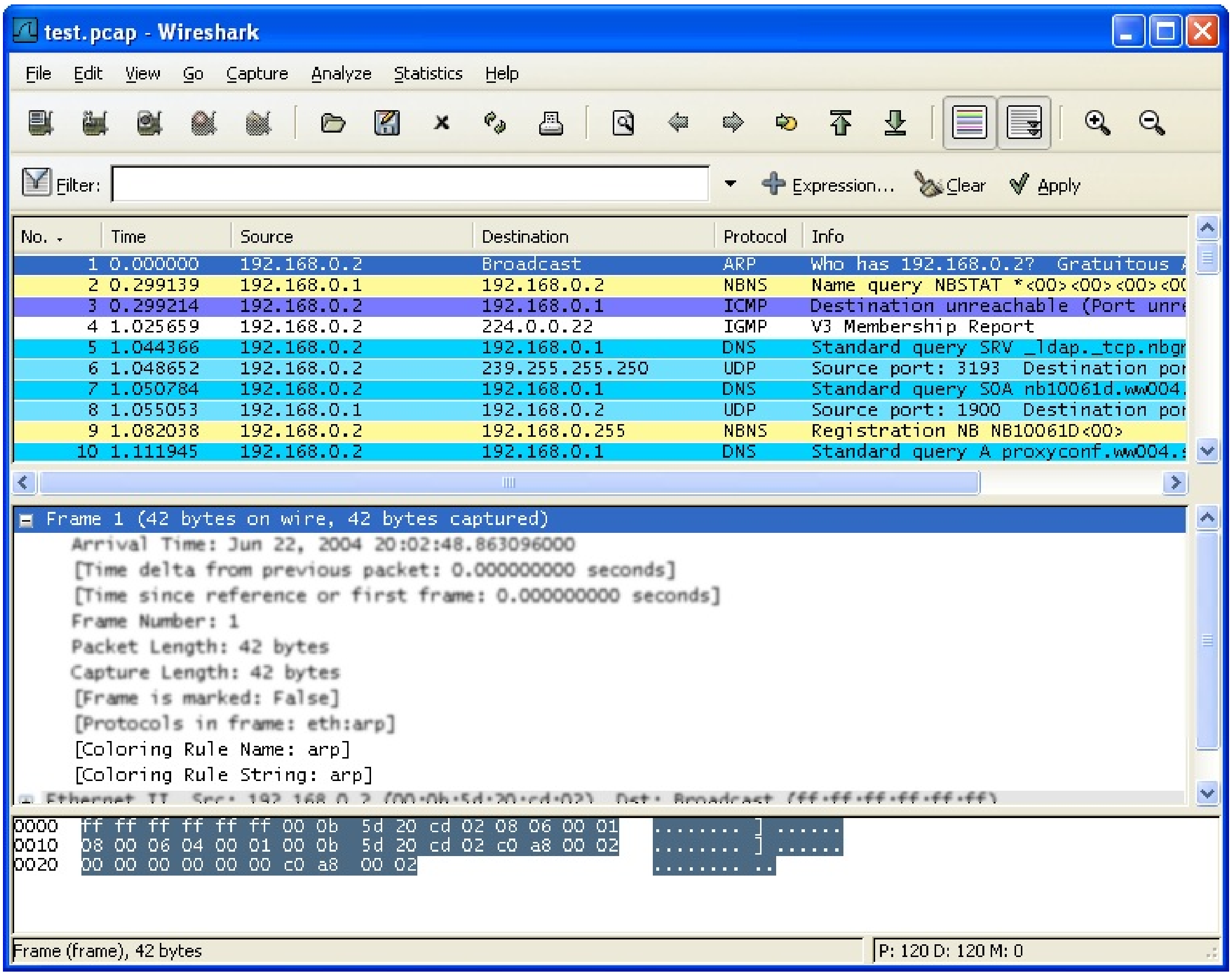
注意

You must select a color in the colorbar next to the colorwheel to load values into the RGB values. Alternatively, you can set the values to select the color you want.

图 9.4 “在Wireshark中使用色彩过滤”显示了默认情况下使用多个色彩过滤器的例子。如果你不太喜欢的话，可以自己随时修改它。

如果你不确定哪个颜色规则会对特定包发生作用，查看[Coloring Rule Name: ...] and [Coloring Rule String: ...] 字段。

图 **9.4.** 在**Wireshark**中使用色彩过滤



9.4. 设置协议解码

用户可以协议如何被解码。[18]

每个协议都有自己的解码器,因此包可能需要使用多个解码器才能完成解码。wireshark会尝试为每个包尝试找到正确的解码器(使用静态"routes"和结构"guessing"),特定的情况有可能会选择错误的解码器。例如，如果你将一个常见协议使用用一个不常见的TCP端口，Wireshark将无法识别它，例如：HTTP协议使用800端口而不是标准80端口。

有两种方式可以控制协议和解码器关联：完全禁止协议解码器，或者临时调用解码器。

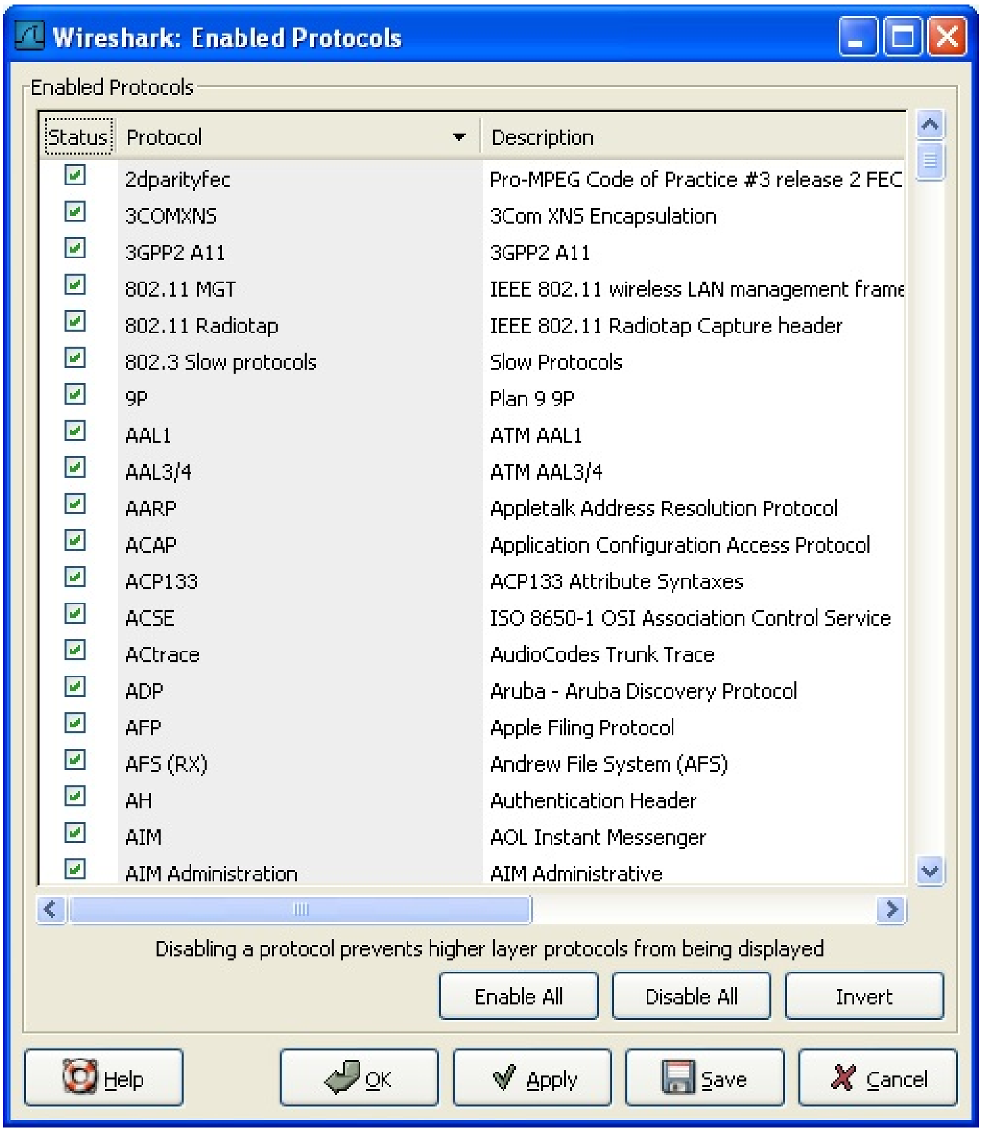
9.4.1. "Enable Protocols"对话框

Eable Protocols对话框可以enable、disable特定的协议，默认情况下是所有协议都enable。如果某个协议被disabled,Wireshark在碰到这个协议时会略过不处理。

注意

禁止某个协议解码会阻止依附该协议的更高层协议显示。例如，假定你禁止了IP协议，选择某个包含Ethernet,IP,TCP和HTTP信息的包。将只会显示以太网信息，IP协议不会显示，基于IP协议的TCP,HTTP协议信息也不会显示。

图 9.5. "Enabled Protocols"对话框



通过点击复选框，或者在协议高亮选中时按空格键可以切换协议enable/disable状态。

警告

必须通过Save按钮保存设置，OK，Apply按钮不会保存设置，关闭Wireshark以后就会丢失设置。

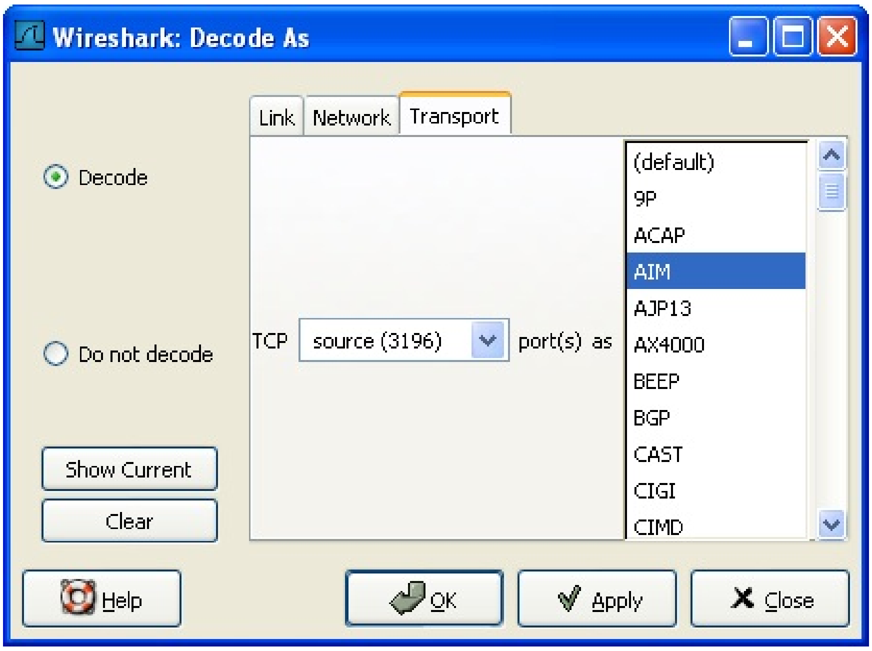
按钮功能介绍

1. **Enable All** 允许列表中所有协议
2. **Disable All** 禁止列表中所有协议
3. **Invert** 切换列表中所有协议的enable/disable状态
4. **OK** 应用当前修改，关闭对话框
5. **Apply** 应用修改，不关闭对话框
6. **Save** 保存当前设置
7. **Cancel** 取消修改，退出对话框

9.4.2. 用户指定解码器

在"packet list"面板，选中包，"Decode As"，打开Decode As对话框，可以临时设置解码器。在协议不使用常见端口时会有所帮助。

图 **9.6. "Decode As"** 对话框



对话框的内容取决于当前选择包的信息。

> 警告
>
> 用户指定解码器不能保存。退出Wireshark以后，这些设置会丢失

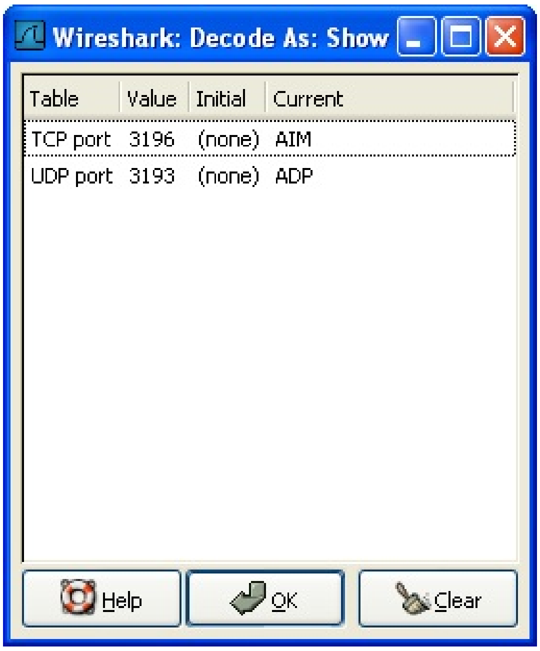
1. **Decode** 使用选择的方式解码。
2. **Do not decode** 不要用选定方式解码。

3. **Link/Network/Transport** 指定使用那个解码器对各网络层进行解码。三个页面中哪个页面可用取决于被选择包的内容。
4. **Show Current** 打开一个对话框显示当前用户已经指定的解码器列表。
5. **OK** 应用当前选定的解码器，关闭对话框。
6. **Apply** 应用当前选定的解码器，保持对话框打开
7. **Cancel** 取消修改，关闭对话框。

9.4.3. 显示用户指定解码器

下面对话框显示了当前用户指定的解码器

图 **9.7. "Decode As: Show"** 对话框



1. **OK** 关闭对话框
2. **Clear** 移除所有解码器

[18] dissector:析像器，应用在光学领域，dissct 解剖，这里姑且把他们翻译成解码器，解码，不过有decode，似乎当作解码有点欠妥。

9.5. 首选项

Wireshark的许多参数可以进行设置。选择"Edit"菜单的"Preferences..."项，打开Preferences对话框即可进行设置。如???所示:默认"User interface"是第一个页面。点击左侧的树状列表中的项目可以打开对应的页面。

注意

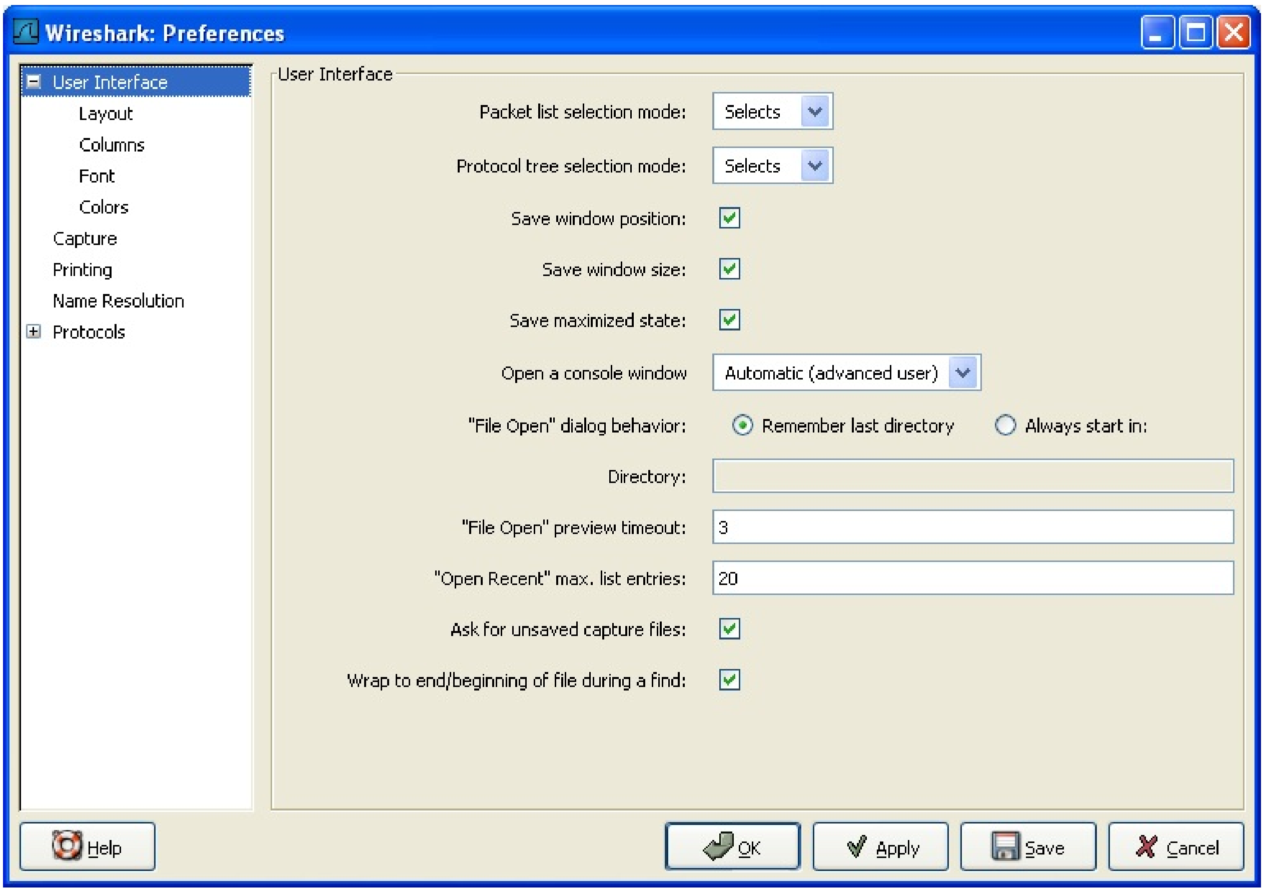
参数设置会频繁追加。想了解关于参数设置的最新介绍，请参考**Wireshark Wiki Preferences** 页:http://wiki.wireshark.org/Preferences.

警告

OK和Apply按钮不会保存设置，你必须点击Save按钮保存设置。

- OK 应用参数设置，关闭对话框
- Apply 应用参数设置，不关闭对话框
- Save 应用参数设置，保存参数设置到硬盘，并且保持对话框打开
- Cancel 重置所有参数设置到最后一次保存状态。

图 **9.8. preferences**对话框



9.6. 用户表编辑器[19]

用户表编辑器是用来管理各种用户自定义参数表。它的主对话框操作方式类似于第 9.3 节 “包色彩显示设置”

[19] 找遍了Wireshark也没看到User table编辑器，版本问题？

9.7. 创建过滤宏

Display Filter Macros是用来创建复杂显示过滤器的快捷方式的工具，例如：定义一个显示过滤宏，名称为tcp_conv 文本为 **((ip.src == $1and ip.dst == $2 and tcp.srcpt == $3 and tcp.dstpt == $4) or (ip.src == $2and ip.dst == $1 and tcp.srcpt == $4 and tcp.dstpt == $3))**，以后你就可以使用**${tcp_conv:10.1.1.2;10.1.1.3;1200;1400}** 替代整个过滤字符串。

显示过滤宏可以通过第 9.6 节 “用户表表”，选择Display Filter Macros 菜单下的View菜单进行管理。用户表有下面两个字段。(好像没有所谓的User table)

name

宏的名称

text

宏的替代文本。使用$1,$2,$3...作为输入参数时。

> 过滤宏的使用说明（译者注）
>
> 首先需要说明的是，实际上在Windows平台GTK2环境下，并没有看到有显示过滤宏功能，可能有的原因有3种：1、0.99.5版本根本没有过滤宏功能；2、我视力不好，没看到，如果是这样，希望谁能帮我找找。3、Windows+GTK2下面没有，其他平台有。
>
> 这里暂且不管有没有，我先按我的理解介绍一下宏的创建使用方法。
>
> 以笔者提到的宏的例子，先说如何创建宏
>
> 1. 定义宏的名称，如范例中的tcp_conv
>
> 1. 定义宏的文本部分，显示过滤宏内容其实和显示过滤器结构上没有本质区别，只是将具体的值换成了参数。，比如例题中第一部分是**ip.src == $1and ip.dst == $2 and tcp.srcpt == $3 and tcp.dstpt == $4**,这里的$1,$2,$3,$4，如果在显示过滤器中，应该是具体的ip地址和端口号，在这里使用了$1,$2,$3,$4,是作为参数。就像定义函数的参数一样，供调用宏时传递参数用的。
>
> 1. 如何使用宏：如例中所示，需要在显示过滤框输入或在过滤表达式编辑器中应用宏，输入宏的格式是${宏名称:参数1;参数2;参数3;....}，参数就是定义宏时的参数的传入值，如例中的**${tcp_conv:10.1.1.2;10.1.1.3;1200;1400}**,tcp_conv是宏名称，10.1.1.2是$1的取值，其他类推。
>
> 再次声明，我装的Wireshark并没有这个功能。希望你们碰到这个共能时能用上。

9.8. Tektronics K12xx/15 RF5 协议表

Tektronix's K12xx/15 rf5文件格式使用helper files(*.stk)验证指定接口的各种协议。Wireshark不能读取stk文件，它使用一个表来识别底层协议。(这句没整明白)

Stk文件协议匹配通过第 9.6 节 “用户表表”来设置,它有两列：

match

a partial match for an stk filename, the first match wins, so if you have a specific case and a general one the specific one must appear first in the list

protos

This is the name of the encapsulating protocol (the lowest layer in the packet data) it can be either just the name of the protocol (e.g. mtp2, eth_witoutfcs, sscf-nni) or the name of the encapsulation protocol and the "application" protocol over it separated by a colon (e.g sscop:sscf-nni, sscop:alcap, sscop:nbap, ...)

9.9. 用户 DLTs 协议表

当一个pcap文件使用用户DLTs (147 to 162)表中的一个时 ,Wireshark使用这个表来识别每个DLT表使用哪个协议。

通过第 9.6 节 “用户表表”管理的DLT表有如下列：

encap

一个用户dlts表

payload_proto

payload(包的最底层协议)协议名称

header_size

如果有header协议(在payload之前)，这个选项告诉Wireshark header的大小。设置为0的话，禁止header protocol.

header_proto

header协议的名称(默认使用"data")

trailer_size

如果有trailer协议的话(追踪协议，在paylod协议之后)，告诉系统它的大小。设置为0表示禁止该协议。

trailer_proto

trailer协议的名称(默认是"data")

9.10. SNMP用户表

Wireshark使用SNMP表验证SNMPv3包的授权并进行揭秘。

该表通过第 9.6 节 “用户表表”进行管理，它包括如下字段。

engine_id

如果输入了engine id,会使用在那些engine id是这些值的包。该字段是一个16进制的字符串，值通常形式为：0102030405

userName

用户名，当一个用户名有多个密码对应不同的SNMP-engines时，第一个匹配的将会被使用。if you need a catch all engine-id (empty) that entry should be the last one.

验证模式

使用什么验证模式,(MD5或者SHA1)

authPassword

授权密码，使用"\xDD"作为非打印字符。一个16进制密码必须输入为"\xDD"形式。例如：16进制密码010203040506就必须输入为'\x01\x02\x03\x04\x05\x06'.

priv_proto

使用的加密算法(DES或AES)

privPassword

私有密钥，使用"\xDD"作为非打印字符。一个16进制密码必须输入为"\xDD"形式。例如：16进制密码010203040506就必须输入为'\x01\x02\x03\x04\x05\x06'.